CMG '87

CONFERENCE PROCEEDINGS

International Conference on Management and Performance Evaluation of Computer Systems

EDITORS
David R. Morley
Jason Shane
Sue Felix-Simpson
Bernie Domanski
John Boelens
Gordon Stauffer
Roberta Terkowitz
Tom Scoumperdis
H. Pat Artis

Orlando
December 7 - 11, 1987

© 1987, The Computer Measurement Group, Inc.

CMG '87 CONFERENCE COMMITTEE

GENERAL CHAIRMAN

David R. Morley
Boole & Babbage, Inc.
510 Oakmead Parkway
Sunnyvale, CA 94086
(408) 735-9550

PROGRAM COMMITTEE

Chairman

Jason Shane
Coldwell Banker Residential Group
Information Services; Suite #200
23046 Avenida de la Carlota
Laguna Hills, CA 92653
(714) 859-5380

Technical

Dr. Bernard Domanski
Bell Communications Research, Inc.
6 Corporate Place
Piscataway, NJ 08854
(201) 699-3838

Sue Felix-Simpson
Morino Associates
8615 Westwood Center Drive
Vienna, VA 22180
(703) 448-0444

John Boelens
Conoco
213 South Park Building
Ponca City, OK 74603
(405) 767-5456

Management

Gordon Stauffer
Morino Associates
8615 Westwood Center Drive
Vienna, VA 22180
(703) 734-9494

Roberta Terkowitz
Amdahl Corporation
4801 Massachusets Avenue, NW,
Suite 600
Washington, DC 20016
(202) 362-7772

Tom Scoumperdis
Boeing Computer Services
P.O. Box 24346
M/S 7A-26
Seattle, WA 98124
(206) 865-5058

Tutorial

H. Pat Artis
Performance Associates, Inc.
2059 Kedge Drive
Vienna, VA 22180
(703) 281-7399

Alan Sherkow
First Wisconsin National Bank
777 E. Wisconsin Avenue, NB
Milwaukee, WI 53202
(414) 796-7070

New Books

James Cook
Data Strategies
9 Aberdeen Street
Irvine, CA 92714
(714) 552-4482

BOF

Larry Kayser
BGS Systems, Inc.
6577 N. Windfield Avenue
Parker, CO 80134
(303) 841-9573

Agenda Editor

Dottie L. Cunningham
Celanese Corporation
P.O. Box 32414
Charlotte, NC 28232
(704) 554-2848

Vendors

Ralph Terkowitz
Washington Post
1150 15th Street, NW
Washington, DC 20071
(202) 334-7196

ADMINISTRATIVE COMMITTEES

Signs

Allen C. Moltz
IDS Financial Services, Inc.
IDS Tower, MS R8-563
Minniapolis, MN 55474
(612) 372-2311

Awards

Mike McElwee
Computer Measurement Services, Inc.
7 Darnby Court
Orinda, CA 94563
(415) 376-6064

Finance

Ian Roome
McDonnel Douglas Aerospace
3855 Lakewood Blvd., MS 1-29
Long Beach, CA 90846
(213) 593-7934

GREAT Program

Karen & Henry Steinhauer
Hewitt Associates
100 Half Day Road
Lincolnshire, IL 60015
(312) 295-5000

Hotel Coordination

Thomas Dennison
CIGNA
P.O. Box 36 UE-8
Somerdale, NJ 08083
(609) 770-6660

Publicity

Bernie Lichtig
Amdahl Corporation
1250 East Arques Avenue
(M/S 116)
Sunnyvale, CA 94086
(408) 746-3398

Referees and Editorial Board

The members of the Program Committee would like to express their thanks to all of the professionals who contributed their time and expertise to the paper referee and editorial review board activities. Each technical and management paper submitted to the conference was reviewed by at least two referees. The referees who assisted the Program Committee in the selection of papers for the conference were:

H. Pat Artis
Thomas Bell
Thomas Beretvas
Mel Boksenbaum
Ethan Bolker
James Bowerman
William Brady
Bobby Brady
James Brennan
Jack Brey
Peter Burghardt
Cliff Burrell
Jim Calaway
James Cheeseman
Kem Clawson
Chuck Comstock
James Cook
Joel Cord
Claris Cottingham
Ann Crocker-Pierson
Paul Crouch
Dottie Cunningham
David Daner
K. Sundar Das
Charles Davis
Brian Day
AnnaBella Deck
Michael Demain
Thomas Dennison
Alan Dipirro
Joe Dirigo
Steven Dodge
Srirama Dronamraju
Gary Dudgoon
Larry Duke
Gene Duncan
Charlene Dykman
William Eagle
Hubert Engelbert
Linda Engstrom
John Fair
William Fairchild
Kit Fechtig
Susan Felix-Simpson
Timothy Fender
Sidney Finehirsh
Dennis Foster
Ellen Friedman
Sarah Fritz
Ronald Galipeau
Gary Garber
Gabe Gewurtz
Roger Gilbert
Geoffrey Goldsmith
Stephen Goldstein
Stephen Goodman
Moshe Gordon
Jo Haggard
Dennis Hahn
Terry Hanks
James Haskins
Lyn Hecker
Joseph Hellerstein
Carson Ho
Charles Hopf
Steacy Housholder
Gilbert Houtekamer
Phillip Howard
Chen-yu Hu
Bruce Ingraham
Philip Ingram
Evangeline Jacobs
Michael Jones
William Judd
Frank Juzwik
Marilyn Kanas
Larry Kayser
Ted Keller
James Kilgore
Julia Klinscek
Robert Klungle
Stephen Knight
Mike Knudson
Walter Kraslawsky
Michael Lee
Thomas Liu
T. Leo Lo
James Loeffler
Douglas MacKinnon
Donald Mack
John Majikes
Alan Maloney
Brian McCay
Robert McClenon
Cal McCracken
Michael McElwee
William McGerty
Tim McGrain
Jeanette McWilliams
Myron Miller
Lawrence Morgan
R. Bruce Morris
Robert Morrison
Gwen Morrison
Brenda Mosley
Bob Mowbray
Robert Mueller
J. William Mullen
Anthony Mungal
Daniel Murphy
Bill Nicholson
Warren Norket
Raymond Nowakoski
Robert Nugen
Randy Numbers
Tim Oliver
Ted Osmer
Charles Padamadan
Thomas Palmer
David Parker
Patricia Perry
John Pilch
Alan Pohli
Linda Potts
Calvin Powell
JoAnn Punch
Henry Ray
Carole Reed
John Reyland
Joseph Richardson
Ellen Robertson
Jerry Rosenberg
Norman Rudikoff
Gary Ruedinger
Gerald Rusthoven
David Sadler
Michael Salsburg
Abe Schoenfeld
Loren Schulte
James Senyszyn
Howard Setkowsky
Jon Shive
David Simone
Cheryl Sluder
Joel Smith
Steve Smith
Gretchen Smith
Connie Smith
Nelson Snyder
Mark Sorkin
Larry Speth
Daniel Squillace
Anthony Standifer
Gordon Stauffer
Gordon Steindel
Jerry Steinhardt
Henry Steinhauer
Joseph Stranges
Mark Streger
Daniel Sumner
William Sundin
Daniel Sutton
Melvin Swanson
Daniel Tartaglia
Steven Tehovnik
Roberta Terkowitz
Ralph Terkowitz
William Tetzlaff
David Thorn
Mike Tomlianovich
Nancy Torok
Joseph Trevaskis
Holly Verbil
Keith VonBargen
Nathan Walsh
TungKung Wang
Robert Waters
Mark Watson
William Westland
Brian Williams
Jerome Wood
Bill Yarnall
Elizabeth Yen
Boris Zibitsker
Raymond Zieverink

After the selection of the papers, the members of the editorial review board were each assigned several papers for a thorough review. During this process, the editors provided the authors detailed suggestions about how to improve the quality and readability of their papers. The members of the editorial review board were:

H. Pat Artis
John Boelens
Bobby Brady
James Cheeseman
Kem Clawson
Chuck Comstock
James Cook
Claris Cottingham
Dottie Cunningham
Bernie Domanski
William Fairchild
Sidney Finehirsh
Dennis Foster
Phillip Howard
Walter Kraslawsky
Douglas MacKinnon
Robert Morrison
Gwen Morrison
Raymond Nowakoski
Randy Numbers
John Pilch
JoAnn Punch
Carole Reed
Gary Ruedinger
Jason Shane
Alan Sherkow
Cheryl Sluder
Gordon Stauffer
Henry Steinhauer
Melvin Swanson
Roberta Terkowitz
Ralph Terkowitz
Holly Verbil

The contributions of the referees and editors are evident in the quality of the conference and these proceedings.

TABLE OF CONTENTS

Technical Papers

Projection Algorithms for IBM DASD Cache Analysis Tool, ... 1
J. Ju & S. Caldwell

A Multiple-Workload Approach to Cache Planning, ... 9
B. McNutt & J.W. Murray

Workload Characterization for Subsystems: A Case Study for IDMS, ... 16
E.M. Friedman

Identification Of Tuning Opportunities Through The Use Of User Behavior Graphs, ... 33
C.E. Hackett

VSAM Specification and Tuning: Optimization of VSAM Performance in Batch and CICS 40
Environments, *F. Bereznay*

In Pursuit of DASD Data Path Alternatives, ... 46
B. Duhl

An Intuitive Approach To Performance Prediction with Application to Workload Management..... 53
in VM SP/HPO, *J. Hellerstein*

A Simple Linear Programming Model For Computation Center Investment Decisions, ... 64
M.J. Merges

A Comparative Study of System/370 CPU's, ... 72
G.E. Houtekamer, W.C. Van Hoboken & A.H. Bonnes

Tightly-Coupled Effect Processor Contention Overhead, ... 79
J.P. Place

Forecasting CPU Utilization in an IMS Environment: A Case Study Using Monte Carlo 81
Techniques, *W.T. Gray & M. Cheng*

Modelling Of A Production IMS System Under A Multi-Function ProcessingArchitecture, ... 87
D. Gilbert

Memory Management For Large IBM Systems, ... 95
M.R. Lee

Modelling Memory Constrained Systems, ... 102
G.E. Houtekamer, W.C. Van Hoboken & A.H. Bonnes

PROFS Capacity Planning, ... 110
A.H. Shub

VM Performance - Top Down That Went Bottom Up, ... 118
M.S. Dickerson

End-User Response Time Measurement - Not The Final Solution, ... 122
P. Futtrup, M Panduro & B Vestermark

Determining End-User Response Time Using NLDM, ... 133
M.W. Yount

General Principles for Performance Oriented Design, ... 138
C.U. Smith

The System Performance Cycle: A Methodology For Architecting Product Performance, ... 145
D. McBride

Expert Systems: Let's Build One Together, ... 151
F.J. Ingrassia

A PROLOG-based Expert System For Tuning MVS/XA, ... 160
B. Domanski

A Flexible Approach To Performance Prediction For Large System Software Development, ... 167
A.M. Long & W.C. Suhler

Sizing New Automated Systems Having Unknown Workloads: A Structured Approach, ... 177
J.L. Gibson, M.C. Ingram & W.M. Smith

An Experiment in the Calculation of Capture Ratios, ... 184
J.F. Temperly & R.J. Hardman

The Use Of Regression Analysis In The Calculation Of Capture Ratios, ... 189
M.F. Kirshin

Cacheing the Tape Library—Tape Reference Patterns and Automated Tape Systems, ... 192
D.A. Kaberon

Analyzing The Performance Characteristics Of 18-Track Tape Technology, ... 201
W.F. Dines & B.A. Cahill

Static vs. Dynamic Management of Consistently Very Active Data Sets, ... 208
M.R. Henley & I. Liu

A Simulation Model for Determining Optimal Freespace Levels on MVS Disk Storage Volumes, ... 217
K.E. Levy

New Measurement Data Sources For MVS/XA and Its Subsystems, ... 222
H.W. Merrill

Expert Systems And MVS Performance Management, ... 233
J. Van Kinsbergen

Expanded Storage Usage And Tuning, ... 237
C. Hanna

Measurement and Analysis of Expanded Storage in IBM 3090-400 Systems, ... 246
J.P. Buzen, J.W. Mullen & A.W. Shum

A Heurustic Modeling Approach To CICS Capacity Planning, ... 252
S.R. Hackenberg

Paging Analysis In An Expanded Storage Environment, ... 256
T. Beretvas

IBM 3090 Processor Expanded Storage Experiments, ... 266
J.P. Pilch & B. Domanski

Is A Picture Always Worth A Thousand Words?, ... 272
D. MacKinnon

An Implementation Of A Graphical Management Information System Using Symphony, ... 279
D. Wilson

The Need for a Network Monitor, ... 282
P.G. Bodley

A Systematic Approach For Identifying Peak Workload Intervals, ... 287
H.P. Artis

Workload-Oriented Performance Analysis, ... 290
S. Samson

On The Optimization of Memory in Large Interactive Environments, ... 296
A.G. Mungal

Estimating The Maximum Throughput Of A Cyber 930 Disk Subsystem, ... 304
J.C. Sutherland & A.W. Tron

Performance Comparison Of Vector Processors On Large Scientific and Business Machines, ... 310
D.S. Lindsay

A Technique For Evaluating The Performance Of Vector Processors, ... 330
D.S. Lindsay

3480 Performance and Capacity Planning, ... 336
T.C. Keller

Transaction Resource Demand, ... 345
J.T. Cheeseman

Workload Characterization Using UNIX System Accounting, ... 351
J.J. Tornheim & R.T. Williams

A Benchmark Comparison of DB2 and the DBC/1012, ... 360
G.C. Steindel & H.G. Madison

Performance Tuning For A VAX-Based Relational Database Application: A Case Study 370
And Lessons Learned, *G.B. Allison*

Empirical Queueing Models of DASD Response Time, ... 390
J.B. Major

Large Capacity DASD: Is Performance Something To Be Afraid Of?, ... 399
B. McNutt

Capacity Planning For A Multi-Processor Environment Supporting CICS Transactions, ... 405
D.T. Gantz, C.M. Gonzalez & M.L. Moore

Model Timeframe Selection And Analysis Of Variance Of An Online Transaction Processing 415
Workload, *D.J. Henry & T.A. Hellner*

Disk Cache Performance Modeling, ... 423
M.A. Salsburg

A Method For Reporting Cached I/O SubSystem Performance, ... 432
N.M. Nearing

Balanced Exponential Approximation Algorithm, ... 441
A. Kumar & M.K. Raghuraman

Sensitivity Analysis Of System Changes Under A Production Environment, ... 446
T.L. Lo

Characterizing UNIX Stream Mechanism Via Pipe Experiments, ... 452
T.P. Lee

Analyzing File System Fragmentation on UNIX Systems, ... 458
B. Samadi

Capacity Analysis Of PBX Systems, ... 464
T.L. Lo & J. Peng

Conceptual Issues In Measuring The Behavior Of A Distributed Real-Time Process-Control 474
Computer System, *J.D. Lawrence*

Management Papers

A Comprehensive Process For Projecting Computer Resources, ... 482
J. Wandzilak & R. Murphy

Capacity Planning As A Business Process, ... 490
J.L. Rosenberg

A Capacity Planning Methodology That Is Destined To Succeed, ... 496
R.T. Lawrence

What We Can Learn From Other Capacity Planners: A Survey of Quantitative Methods, ... 504
B.A. Ingraham

Managing Departmental Mainframe Demand Through Shared Accountability, ... 508
K.C. Jones

Management Reporting: An Approach To Availability And Service Reporting, ... 515
M.B. Marcus

Quality Improvement In Capacity Management, ... 518
B. Wessner

A Performance/Capacity Data Base For Long Term Trends And Studies, ... 522
H.E. Jack, K. Goldenberg & R. Williams

Financial Considerations In Capacity Planning, ... 531
J.L. Rosenberg

Capacity Budgets And Financial Budgets: Relating Them To Each Other Through Resource 538
Accounting, *J.W. Richardson*

Experience in Cost Allocation, ... 543
J.J. Stancavish

Chargeback vs Cost Accounting: The Appropriate Emphasis, ... 546
G.J. Gillen & B.F. Henry

Experiences With Performance Engineering, ... 549
T.E. Bell, A.M. Falk & T.M. Thorley

Service Level Agreements "Good Fences Make Good Neighbors", ... 553
G.W. Miller

Capacity Planning in the Age of Competitive Advantage Computing, ... 561
L.D. Lipner & R.P. Goldberg

Structured MIS Planning, ... 566
V.J. Salminen

Who Goes There? A Dialogue Of Questions And Answers About Benign Hacking, ... 569
D.F. Stevens

Real World Performance Measurements For Unix Based Systems, ... 574
N. Nelson

Storage Performance, An Eight Year Outlook, ... 579
S. Goldstein

Cache Management in a Hostile Environment, ... 586
R.C. Wells

Capacity Management For Laser Optical Devices, ... 591
D.E. Sarna

Magnetic Cartridge Tape Configuration Management, ... 596
A.R. Guerrero

End-User Performance Measurements: An Automated PC Approach, ... 599
G.K. Martin Flickinger

Incorporating Data Networks Into A Performance And Capacity Planning Structure — 606
A Case Study, *H.L. Smith & D.D. Wilson*

Some Experiences of Data Centre Relocation, ... 608
B. Day

The Role Of Capacity Planning In Consolidations, Acquisitions and Mergers, ... 611
H.E. Verbil

Can Service Units Be Used For Uniform Chargebacks?, ... 617
C.V. Claunch

A Fully Automatic Analytic Approach to Budget-constrained System Upgrade, ... 620
S.T. Chanson & A.S. Wong

Management Tutorials

Establishing Useful Business Relationships, ... 630
L.E. Allen

A Systems Engineering Look at SAA, ... 636
J.R. Cook

DASD Management: Let's Talk Money, ... 651
G.H. Parker

Workload Characterization And Forecasting For A Large Commercial Environment, ... 655
G.W. Miller

A Strategy For Improving IO Subsystem Design Using Cost Per IO Analysis, ... 672
A.L. Jones

Report On The Large System Storage Hierarchy, ... 685
J.W. Percell & F.G. Moore

Bullmoose Does TA: A Manager's Guide to Transactional Analysis, ... 698
J.A. Stanton

An Introduction to Business Element Forecasting, ... 703
J.R. Bowerman

A Practical Use Of Natural Forecasting Units, ... 710
J.M. Reyland

After The Study Is Over, ... 714
C.A. Cooley

Computer Acquisition After The Tax Reform Act Of 1986: Exploring The Opportunities, ... 719
S.P. Amembal & S. Halladay

Incorporating Expert Systems Technology Into Software Performance Engineering, ... 720
D.J. Weishar

The Goldilocks Syndrome, ... 723
M. Berry

Introductory Tutorials

Systems Performance Management and Capacity Planning, ... 725
B. Domanski & J.P. Pilch

Fundamentals Of Statistical Workload Forecasting, ... 733
H.P. Artis

A Capacity Planning/Queueing Theory Primer, ... 734
E.D. Bolker

High Performance Processor Design, ... 736
B. McCormack, J. Michlin, L. Brisson & A. Martin

Implementing Network Capacity Planning, ... 741
T. Olberding

The Theory And Practice Of I/O Tuning, ... 749
R. Olcott

Online Data Base vs. Raw System Data, ... 761
R.H. Denis

Methodology To Develop A Tool For Monitoring CPU Usage By Applications Running Under VM, ... 764
R.D. Deese & H.H. Ghannad

CICS Performance and Workload Reports, ... 767
C.L. Comstock

An Introduction To Program-To-Program Communication And Peer-To-Peer Networking 774
In An IBM SNA Environment, *T.W. McDonald*

A Real-Life "How-To-Do-It" Guide To Modelling, ... 787
M.L. Taylor & M. McElwee

Analytic Modeling for the Non-Analytic Modeler, ... 790
B. Sannerud

VM/HPO Performance Measurements, ... 795
M.S. Streger

Capacity Planning By The KISS Method, ... 797
L. Kayser

Reporting Availability At The Job Step Level, ... 801
R.J. Scott

Advanced Tutorials

What Models Do And What Models Don't, ... 807
A.I. Levy

Accounting for Workload Variation in Interval Selection and Analysis, ... 808
B.E. Williams

Tools And Methods Available For In-Depth VM Performance Analysis, ... 816
D.C. Christopher

CICS and DB2: A Marriage Made in Heaven?, ... 822
J.G. Shane & J.R. Cook

Modelling CICS Systems: The Performance Impact Of MRO Implementations And Data 828
Base Considerations, *E.M. Friedman*

P. S. I Love You. Signed TSO, ... 835
B.A. Lichtig, R.W. Martindale & T.L. Parmelee

A Capacity Planning Methodology For Data Networks, ... 858
J.E. Harmon

Netview and Netview/PC Tutorial, ... 865
K. Terplan

A Survey of Performance Applications of The Teleprocessing Network Simulator (TPNS), ... 880
R.C. Burke

I/O Tuning In The MVS/XA Environment, ... 889
R. Olcott

From Simple Regression To Analysis Of Variance And Beyond—Painlessly, ... 901
A.C. Pool

High Performance I/O Subsystems, ... 916
A.G. Mungal

Paging And Swapping Interactions, ... 929
J.P. Buzen, J.W. Mullen & A.W. Shum

Performance Evaluation Of Future System Designs, ... 936
J.M. Quernemoen, M.A. Fahland, B.G. LaPlante & T.L. Nelson

3090-Class Processor Architectures and Workload Performance Levels, ... 947
G. Keeney & G. Simmons

Coupled Systems For Performance, ... 949
E.I. Cohen

New Books

A Computer and Communications Network Performance Analysis Primer, ... 959
B.W. Stuck & E. Arthurs

A Close Look At MVS Systems: Mechanisms, Performance And Security, ... 962
R. Paans

CICS - Designing For Performance, ... 970
E.P. Emanuel

Panel Discussions

Rules of Thumb: Science or Science Fiction, ... 981
H. Zimmer

Good Tools And Models Are Not Enough: On The Necessity For Bi-Directional Communication, ... 988
E.E. Robertson & D.L. Cunningham

Senior Management Expectations of Capacity Planning, ... 989
J.A. Stanton

Successful Software Performance Engineering, ... 991
C.U. Smith

PROJECTION ALGORITHMS FOR IBM DASD CACHE ANALYSIS TOOL

Jean Ju and Stan Caldwell
International Business Machines Corporation

ABSTRACT

DASD Cache Analysis Tool (DCAT) is an IBM internal tool which models and projects the performance of both cached and uncached I/O subsystems. The current version covers MVS/XA, MVS/SP, VM/SP, and VSE environments. For MVS/XA, DCAT supports up to four hosts. DCAT provides performance information based on workload parameters, configuration, and device characteristics. Workload parameters are input by the user, calculated from system monitor data, or derived from migration sources. This article discusses projection algorithms used in DCAT. Comparisons of model results with benchmark measurements are also presented.

1. INTRODUCTION

DASD Cache Analysis Tool (DCAT) is an IBM internal tool which provides the ability to model I/O subsystem performance and to project the performance impact of DASD and cache migrations. This paper will cover the I/O subsystem model concentrating on algorithms used for projection.

First, a brief introduction of the general background of DCAT will be given. Input to the DCAT model includes device models, workload parameters, and configuration. DCAT supports devices 3350, 3370, 3375, 3380, 3380D and 3380E attached to control unit 3880 model 3. DCAT also supports the 3380 family attached to cached control units, 3880 models 13 and 23. Workload parameters are defined by users, derived from system monitor data, or calculated as part of the projection process. The configuration information is required in calculating the impact of I/O subsystem resource contention. The configuration is defined by specifying which devices belong to which LCU (Logical Control Unit) and what channel paths are available to each LCU.

DCAT provides a calibration capability which accepts system monitor data as input, and produces workload parameters to be used for projections and "what if" analysis. Major workload parameters include the activity rates, seek times, search times, and transfer sizes by device. A report is provided which compares the modelled results for that workload with the associated monitor data. Currently, DCAT supports monitor data for MVS/XA(RMF), MVS/SP(RMF), VM(MAP) and VSE(PT). The derived workload parameters can be used as a base for analysis using the projection options.

The user is typically interested in the performance impact given one or more of the following scenarios:

- an increase in the throughput requirement
- migration to a cached control unit
- migration to a new device type
- combination of multiple volumes onto a single device
- the addition of new devices to an existing string or control unit
- an increase in cache size
- migration from MVS/SP to MVS/XA

The projection option allows analysis of I/O growth projection and device migration. Sub-options allow the user to select various source volume combinations and mappings for the migration. Growth projections can be evaluated overall or for a subset of volumes based on user defined groupings called application groups in DCAT. The cache projection option focuses on cache migrations. The needed cache parameters can be input by the user or selected from typical workloads. This option also provides the ability to do sensitivity analysis on hit ratio or cache size.

This article will provide readers with a general understanding of DCAT so that they can discuss DCAT analysis provided by their IBM system engineers or adopt a DCAT approach for their own use.

2. SUBSYSTEM MODEL

Here we present an analytic model for predicting I/O subsystem performance. The model focuses on a simple form of channel program, such as this BDAM (Basic Direct Access Method) chain of CKD (Count Key Data) channel commands:

Seek
Set File Mask
Set Sector
Search ID Equal
Read

An equivalent ECKD (Extended Count Key Data) version has channel commands: Define Extent, Locate Record, and Read Data. To avoid unnecessary complexity, we will illustrate the model for a subsystem containing one string of 3380D attached to a 3880 model 3 control unit in an MVS/XA environment.

The model is based on queueing theory (Ref. 2). The channel program is treated as a request and the Unit Control Block (UCB) is viewed as a server. If an arriving request finds the UCB busy, then it will be placed on the UCB queue. The time it waits here for the UCB is reported by RMF (Resource Measurement Facility) as IOSQ (IOS Queue) time. After a channel program gets the UCB, it tries to access the device. If any component of the path to the device is busy, or if the device itself is busy, there is additional delay which is reported by RMF as PEND time. UCB service time is the sum of the physical device service time and PEND time. The physical device service time is broken into connect and disconnect time components, and reported by RMF as CONN time and DISC time respectively. UCB utilization is the product of the activity rate and UCB service time. Further breakdown of physical service time will be necessary for characterizing device and workload parameters. The components of physical device service time dealt with in the model are:

Seek: Time spent moving the access mechanism to the proper cylinder and track.
Latency: Time from end of seek until reconnect target sector.
RPS Delay: Lost revolutions due to failure in reconnecting.
Overhead: Portion of control unit overhead for the above BDAM channel program which is not overlapped with other service time components.
Transfer: Data transfer time for read/write commands.

Search: Time for repeated comparisons of record ID's or keys. In DCAT, the search time includes the pure search time as well as additional connect time spent handling more complicated channel programs.

Following is the list of terms for model variables which will be used in discussing DCAT's algorithms:

CONN: connect time
DISC: disconnect time
IOSQ: IOS queuing delay
LAT: latency
OVHD: non-overlapped control unit overhead
PEND: pend time
RESP: response time
RPSD: RPS (Rotational Position Sensing) miss delay
SEEK: seek time
SRCH: search time
STPDV: physical device service time
STUCB: UCB service time
XFER: data transfer time

The model can be summarized by the following formulas:

```
CONN  = OVHD  + SRCH + XFER
DISC  = SEEK  + LAT  + RPSD
STPDV = CONN  + DISC
STUCB = STPDV + PEND
RESP  = STUCB + IOSQ
```

The seek, transfer, and search times are functions of device characteristics and workload patterns. Half of the device rotation time represents the average latency. The overhead value is based on control unit measurements for the simple BDAM channel program. RPS miss delay and Pend time are calculated as functions of path utilization. RPS miss delay will be discussed in more detail in the next sub-section. It should be noted that for I/O operations on cached devices, read hits are satisfied directly out of the cache and have no disconnect time.

Pend time is relatively small for the single host case. It only involves delays associated with path busy at I/O initiation and is a function of the connect time and utilizations of storage directors and channels. For multiple host cases, when a device is shared, there can be a large Pend time due to the device being busy with another host. Benchmarks have shown that in many cases one host continues to monopolize a device until its queue of outstanding requests has been emptied. DCAT models this device level delay as a function of the device and UCB service times and utilizations. The model calculates IOSQ time by applying an M/G/1 queueing formula (Ref. 2) to the UCB service time and the UCB utilization. We would now like to give special attention to estimation of RPS miss delay.

2.1 RPS MISS DELAY

In Figure 1, two channel paths from the same MVS/XA host are connected with two storage directors in a 3880 control unit model 3. Sixteen 3380D devices can reconnect through either of two storage director-channel paths. Consider the case that device k is attempting to reconnect through storage director j and back to channel i. If only one LCU is connected to the pair of channels, the only opportunity for an RPS miss is when both storage directors are busy with other devices. There is no additional contention introduced by channels since a free storage director guarantees a free channel.

The RPS miss probability is calculated for each device. Let PSDM(j,k) be the conditional probability of an RPS miss in getting storage director j given that device k is not connected to storage director j. Then,

$$PSDM(j,k) = (USDj - USDjDVk) / (1 - USDjDVk)$$

where USDj = utilization of the jth storage director,
USDjDVk = utilization of storage director j by device k.

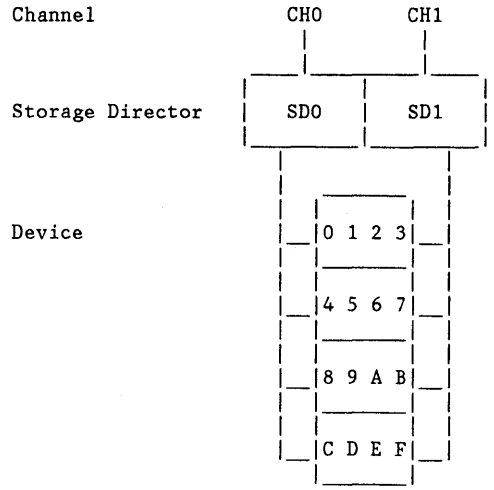


Figure 1. Subsystem Configuration for 3380D's Attached to 3880

The numerator represents the utilization of a path associated with other devices on this LCU and the denominator conditions upon this particular device not being currently connected to the storage director. Balanced loads are assumed for the two storage directors. The M/M/2 queueing formula (Ref. 2) is used to reflect MVS/XA's dynamic reconnect capability, which allows either of the two paths to be used in getting back to the host. Let PM be the conditional miss probability of an average reconnect attempt and U be total "conditional utilization" to two servers, then

$$PM = M/M/2 \text{ queuing of } PSDM$$
$$= 1 - ((1 + U) / (1 + U + (U \times U) / (2 - U)))$$

where U = 2 x PSDM

This equation will be applied to each device. In multiple host environments, the storage director utilization is the total utilization from all hosts since interference with reconnection might come from any source.

Configurations which involve two or more LCU's sharing two channels require that channel contention be considered. Assume that device k belongs to LCU m. Let PCHM(i,m) be the conditional probability of an RPS miss in getting channel i given that a storage director in LCU m is available, then

$$PCHM(i,m) = (UCHi - UCHiLCUm) / (1 - UCHiLCUm)$$

where UCHi = utilization of the ith channel,
and UCHiLCUm = utilization of channel i by LCU m.

Apply the M/M/2 queueing formula to PSDM and PCHM and compute the following probabilities for n = 0, 1, and 2.

PSDB(n) = conditional probability of n storage directors (in this LCU) busy with other devices, given that device k is not connected
PCHB(n) = conditional probability of n channels busy with other LCU's, given that a storage director is available in LCU m.

The probability of a miss for a storage director-channel pair is a function of PSDB's and PCHB's. A matrix of coefficients is applied to the product of PSDB and PCHB, since the availabilities of storage directors and channels

are not independent. The probability of miss can be expressed as

```
      PM   = Sum of (PSDB(i) x PCHB(j) x COEF(i,j))
             for 0 =< i,j =< 2,
where COEF =  0  0  1
              0 .5  1
              1  1  1
```

For multiple host cases, these formulas will use total device and total storage director utilizations and the individual channel utilizations from each host. Remember that PM is the miss probability for an individual reconnect attempt; there can be more than one miss for a single disconnect. The expected number of misses per SIO will be PM / (1 - PM). This can be viewed as the rate of failures per success. The model assumes there is one successful reconnect per SIO; so it is the miss rate per SIO. Finally, the RPS miss delay is

RPSD = rotation time x PM / (1 - PM).

2.2 RMF CALIBRATION

The subsystem model described above assumes components of the response time are available or can be calculated according to DCAT's algorithms. If a user starts with RMF data and runs through the DCAT calibration option, he will end up with a file containing workload parameters corresponding to his RMF data. Those workload parameters can be used to derive components of the response time. DCAT then compares the modelled response and service times with RMF data and reports the calibration ratios.

Let us take a closer look at RMF data for the MVS/XA environment. RMF reports physical device service time in two parts: disconnect time (DISC) and connect time (CONN). The formulas for disconnect and connect times which will be used to describe the calibration process are repeated below:

```
STPDV = DISC + CONN
DISC  = SEEK + LAT  + RPSD
CONN  = OVHD + SRCH + XFER
```

For uncached control units, the process of calibration can be explained simply. Given RMF data and configuration, the workload parameters SEEK, SRCH, and XFER can be estimated as follows:

DISC: First calculate RPSD based on configuration and path utilizations. Derive SEEK as follows:
SEEK = DISC - LAT - RPSD.

CONN: Split (CONN - OVHD) into SRCH and XFER based on historical ratios of SRCH and XFER.

When a configuration includes a cached control unit, multiple channel program types are considered when determining components for disconnect and connect times. For example, a read channel program which finds its data in cache will not require device access and will contribute zero time to the average disconnect reported by RMF. Also the SRCH time for a search in cache will be less than the SRCH time for a search requiring device rotation. Therefore, the rules are similar to the uncached case given above but are lengthier because that multiple channel program types are used and each channel program type may contribute different amounts to disconnect and connect times. The implementation of the above process involves iteration and requires input of cache statistics such as the ratio of reads to writes and the fraction of reads which are satisfied in the cache.

For MVS/SP, VM, and VSE environments, the monitor data does not break down the device service time into disconnect and connect components. DCAT uses the channel service time, which can be derived from the channel utilization, as an estimate of connect time for each device. The disconnect time is obtained by subtracting the connect time from device service time, and then the procedure discussed above is applied.

Once the workload parameters have been generated, the user can use these parameters in running the regular model to do 'what if' analysis. There are some limitations on this calibration scheme. It focuses on a simple form of channel program as described in the beginning of this section. Some applications generate channel programs which do not match the simple pattern; for example, there may be no Set Sector or Locate command. Devices which include a significant amount of "unusual" activity will not calibrate. In general, the disconnect and connect times have lower bounds: a half of the rotation time and the channel overhead, respectively. Any volume used primarily by applications which generate smaller disconnect or connect time than these limits will have trouble calibrating. Typically the number of volumes and fraction of I/O which exhibit this behavior is small and the overall subsystem calibration remains good.

Experience shows that in MVS/XA environments, the model calibrates to within 1% of the RMF data. In non-MVS/XA environments, since connect times are not provided for each device and must be estimated, calibration within 5% to 10% of the measured data is typical.

3. MIGRATION AND SEEK PROJECTION ALGORITHM

We will cover general algorithms for growth and migration first. Then we move on to the seek projection algorithm, which is a major issue for device migration.

In the growth scenario, those variables which change are prefaced by 'G'; other variables remain the same. The growth that is being modelled is an increase in I/O rate for all or a subset of the volumes. The migration scenario assumes that a set of volumes are being moved from their current devices to some new device type. This move may involve folding several volumes from small devices onto a single larger capacity device. In this case, variables prefaced by 'W' indicate that the variable will be a simple weighted average of the original volume variables. A prefix of 'M' indicates that this variable will be affected in other ways by the migration.

3.1 WORKLOAD GROWTH

For growth projection, we have disconnect, connect, and pend times as below:

GDISC: RPS miss delay will be grown with the activity rate and utilizations.
GDISC = SEEK + LAT + GRPSD

GCONN: The average connect time for each SIO remains the same.
GCONN = CONN

GPEND: Pend time will be grown with the activity rate and utilizations.

After the variables are changed as described, DCAT will output the response time, service time, and utilizations based on the increased I/O rate. Results are provided at an overall summary level by volume and by application group since an increase in I/O rate for one application can have a negative effect on response time for another.

3.2 DEVICE MIGRATION

Now consider a group of source volumes that are to be migrated to a new group of target devices. For each target device, disconnect, connect, and pend times can be determined by the following rules:

MDISC: Calculate MSEEK according to the seek projection algorithm. Calculate MRPSD according to new configuration, utilizations, and activity rates for target devices. Use half of the rotation time of the target device as the new latency.
MDISC = MSEEK + MLAT + MRPSD

MCONN: Assign the weighted averages of source search times and source transfer times as the search and transfer times of the target device. The overhead may change if the migration involves a new control unit or micro code load.
MCONN = MOVHD + WSRCH + WXFER

MPEND: Pend time is a function of new configuration, service time, utilizations, and activity rates.

3.3 SEEK PROJECTION

The average seek time of the target device will be determined by the pattern of seeks on source volumes and additional arm stealing among volumes introduced by the migration. Arm stealing occurs when two or more datasets are busy in a small time window. We will be able to estimate the target arm stealing rate if we can establish a dataset busy distribution for the target device. Assume for the moment that probabilities of zero, one, and two or more datasets busy are known for each source volume in a small window. The target device has no dataset busy if and only if all the source volumes have no datasets busy. The target device has one dataset busy if and only if exactly one source volume has one dataset busy. The hidden assumption here is that the activities of the source volumes are independent. We can express the above algorithm mathematically by means of the following notation:

A0(i): Probability of no dataset busy in source device i

A1(i): Probability of one dataset busy in source device i

A2(i): Probability of more than one dataset busy in source device i

B0: Probability of no dataset busy in the target device

B1: Probability of one dataset busy in the target device

B2: Probability of more than one dataset busy in the target device

Then the relationship among Ai's and Bi's for n-to-1 mapping follows:

$$
\begin{aligned}
B0 &= A0(1) \times A0(2) \times A0(3) \times .. \times A0(n-1) \times A0(n) \\
B1 &= A1(1) \times A0(2) \times A0(3) \times .. \times A0(n-1) \times A0(n) \\
&\quad + A0(1) \times A1(2) \times A0(3) \times .. \times A0(n-1) \times A0(n) \\
&\quad + A0(1) \times A0(2) \times A1(3) \times .. \times A0(n-1) \times A0(n) \\
&\quad + ... \\
&\quad + A0(1) \times A0(2) \times A0(3) \times .. \times A1(n-1) \times A0(n) \\
&\quad + A0(1) \times A0(2) \times A0(3) \times .. \times A0(n-1) \times A1(n) \\
B2 &\text{ satisfies } B0 + B1 + B2 = 1
\end{aligned} \qquad (1)
$$

The following regression formulas which are used to get probabilities of dataset busy and average seek times are extensions of techniques described in Ref. 3. The trace data used there were collected during peak-load time intervals at two large installations.

Let

ACTR: Activity rate

K1,K2,K3: Coefficients

Tr: Nominal seek time

Ts: Average seek time within a single dataset

THETA: Arm-stealing rate

Then

$$
\begin{aligned}
SEEK &= (1 - THETA) \times Ts + THETA \times Tr \\
THETA &= (K1 + K2 \times A2) / ACTR \\
ACTR/K3 &= A1 + A2 \\
1 &= A0 + A1 + A2
\end{aligned} \qquad (2)
$$

where Ts/Tr = 0.1, K1 = 0.1, K2 = 30, and K3 = 30

The first formula means the average seek time can be estimated by the the nominal seek time when arm stealing occurs and by the average seek time within a single dataset when there is no arm stealing. By applying these regression formulas, DCAT can derive probabilities A0(i), A1(i), and A2(i) for each source volume i. Then it uses equation set (1) to gain dataset busy probabilities B0, B1, and B2 for the target device. Finally, it applies equation set (2) again to derive the average seek time for the target device. Some modifications are needed to assure that all probabilities stay between zero and one.

$$0 =< A0, A1, A2, (A1+A2), \text{ and } THETA =< 1.$$

Also, modifications are applied to account for the "short-stroke" effect when there is one dominant source volume. Further discussion on the short stroke effect can be found in Ref. 1.

2-to-1 mapping:
Modified Target Seek = 0.90 x Target Seek
3-to-1 mapping:
Modified Target Seek = 0.84 x Target Seek
4-to-1 mapping:
Modified Target Seek = 0.81 x Target Seek

This seek projection algorithm has been validated using simulation runs. The simulation looks at trace data to determine the pattern of seeks and seek distance for each I/O. The seek time associated with these seek distances is based on actual device measured seek curves, and an average time for each volume is generated. To simulate a migration to 3380E from 3380 (2-to-1), the assumption is made that data from the first 3380 occupies cylinders 0--884 and data from the second is on cylinders 885--1769. The traces for the two volumes are merged and a new seek sequence is generated. The average time for this new migrated sequence can then be compared with what is predicted by DCAT's algorithm. In general, the simulation is somewhat pessimistic since the possible data compression of empty space on the volumes is not being accounted for. Also, the new larger volume would have only one VTOC (Volume Table Of Contents) and hence would not have the arm stealing between VTOC's which is simulated.

Simulations were run with customer I/O trace data and set up to cover various n-to-1 mappings and two different foldings -- slide and random. It is convenient to define slide folding by an example. Say there are 12 devices being numbered according to the order of their activity rates. For 2-to-1 mapping, slide folding will put device pairs (1,7), (2,8),..., (6,12) as target devices. For 3-to-1 mapping, it has sets (1,5,9), (2,6,10), (3,7,11), and (4,8,12). The complete mappings between source and target devices are illustrated as below:

Slide Folding:	source devices ordered by activity rates						
2-to-1 mapping	---------->	1	2	3	4	5	6
	---------->	7	8	9	10	11	12
		:	:	:	:	:	:
	Target Device	1	2	3	4	5	6
		:	:	:	:		
3-to-1 mapping	---------->	1	2	3	4		
	---------->	5	6	7	8		
	---------->	9	10	11	12		

Slide folding is used in DCAT when the default migration option is selected. It will provide a reproducible and relatively conservative distribution of activity among target devices. The optimal folding for the 2-to-1 mapping for the above example will put device pairs (1,12), (2,11), ..., (6,7) as target devices. DCAT does not pick

this optimal folding as the default since the folding that is optimal overall is unlikely to be observed during any one particular interval, and was therefore thought to be too optimistic. Random folding corresponds to a user assigning the migration mapping with no information or knowledge of device activity. Since users can normally identify their busier volumes, this random approach was felt to be too pessimistic. However, the seek projection algorithm was checked for random folding to ensure the coverage of this extreme case.

Comparisons of analytic seek projections versus simulation runs are plotted in Figures 2 to 6. The first three figures show comparisons for 2-to-1, 3-to-1, and 4-to-1 mappings for the slide folding. The next two figures show all mappings for slide folding and for random folding. The scale is normalized by the nominal seek of the target device. The points on the line from (0,0) to (1,1) correspond to the analytic and simulation results being identical. All five plots demonstrate reasonably good matches between the analytic model and the simulation.

We have studied other approaches for the seek projection algorithm which use the average of all possible combinations of 2-to-1 mappings. Compared to those approaches, this algorithm is independent of the order of source devices and saves computing time. In addition to the simulation validation, measurements have been used to do an overall validation of the device migration scenario, and are presented in section 5. We believe the seek projection algorithm presented in this section to be a reasonably fast and accurate approach for both slide and random foldings.

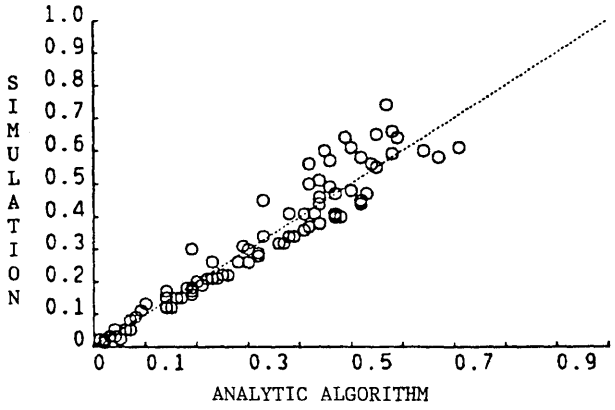


FIGURE 2. SEEK PROJECTION VALIDATION, 2-TO-1 MAPPING

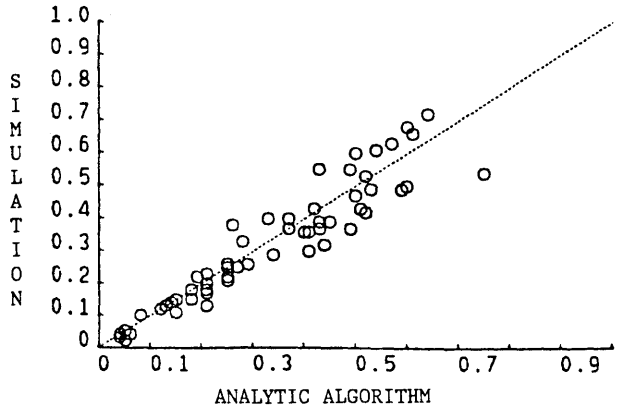


FIGURE 3. SEEK PROJECTION VALIDATION, 3-TO-1 MAPPING

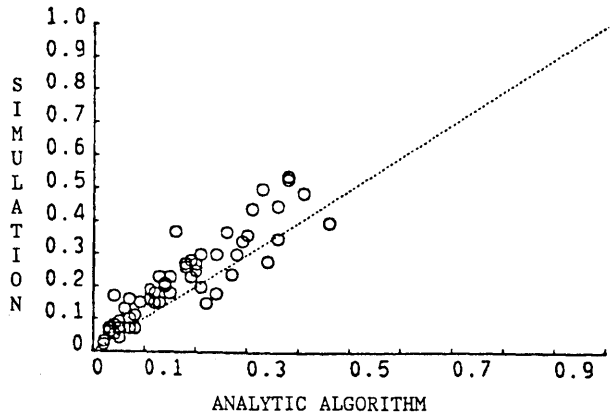


FIGURE 4. SEEK PROJECTION VALIDATION, 4-TO-1 MAPPING

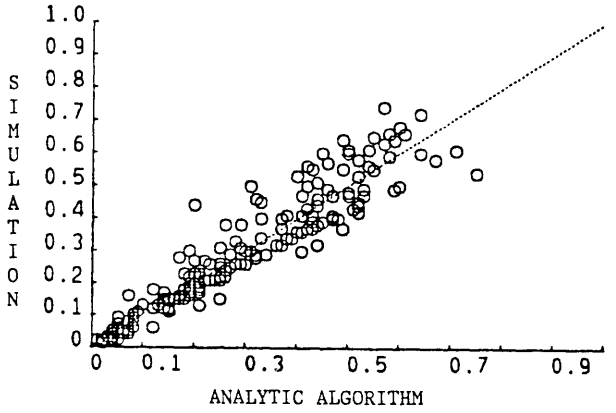


FIGURE 5. SEEK PROJECTION VALIDATION, SLIDE FOLDING

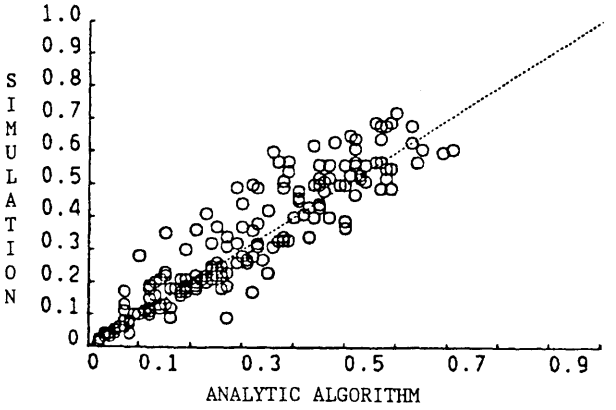


FIGURE 6. SEEK PROJECTION VALIDATION, RANDOM FOLDING

4. CACHE PROJECTION AND ITS ALGORITHMS

4.1 UNCACHED TO CACHED CONTROL UNITS

Up to this point we have centered the discussion on how DCAT models the uncached devices. The DCAT model for cached devices is similar but where the uncached activity is reduced to one simple BDAM channel program, the cached activity is reduced to multiple channel program types. The four most important channel program types are: read hit, read miss, write hit, and write miss. For the 3880 model 23, all but the read hit channel program type in-

clude access to the physical device, which means that the service time includes seek, latency, and RPS miss delay.

For modeling a cached control unit, determining the allocation of a device's throughput among the various channel program types is extremely important. For example, a Start I/O which only reads 4K data bytes and finds the data in cache storage can have a response time in the 3 to 4 millisecond range since no device activity is involved. Also contributing to the low service time for read hits is a faster search in cache versus a comparable search time on a physical device. The same SIO occurring as a read miss would be similar to an uncached SIO and might take 25 to 30 milliseconds. Therefore, two additional important workload parameters for each cached device are read-to-write ratio (RWRATIO) and read hit ratio (READHIT).

The cache projection portion of DCAT allows a user to project from uncached to cached control units, project from one average hit ratio to another, and project from one cache size to another.

Projecting from uncached to cached configurations requires that DCAT query the user for the needed cache parameters. The user has the option of deriving the cache parameters for the system being modeled or letting DCAT default the parameters. The most accurate method is for the user to derive the cache parameters from data collected on the system being being modeled. The Cache Analysis Aid (CAA) tool is available for the user which analyze trace data to determine the read-to-write ratios and the read hit ratios for different cache sizes. For those studies not requiring this level of accuracy, DCAT can provide "canned" cache parameters based on measurements previously done at installations with similar workloads.

4.2 HIT RATIO PROJECTION

Projecting from one average hit ratio to another is done by giving the user the current average hit ratio for a control unit and allowing the specification of a different average. DCAT takes the new average hit ratio and adjusts the individual device hit ratios before running the model.

When developing the algorithm to determine new device hit ratios, many things had to be considered. For example, if the target average hit ratio is larger than the original average hit ratio, then it was felt that all the device hit ratios would probably increase. Also it was felt that a device with a high hit ratio would probably not increase as much as a device with a low hit ratio. The above two conditions had to hold in reverse when the target average hit ratio is less than the original average hit ratio.

Another thing to consider was, if the target average is 0, 1 or equal to the original average, then the individual device hit ratios need to be 0, 1, or equal to the original device hit ratios respectively. In addition, all hit ratios are bounded by 0 and 1. Part of the original challenge was to find an algorithm which was reversible such that one could project to a target average hit ratio and reverse the process arriving at identical device hit ratios as at the starting point. The algorithm presented does have the property of reversibility until any device hit ratio starts at a boundary or is projected to a boundary. In order to provide the property of reversibility, the original device hit ratios would have to be kept. The property of reversibility was not important enough to justify the additional storage required.

The algorithm is as follows:

HR(i) - the read hit ratio of device i, which is the current setting

AHR - average read hit ratio of all devices on a control unit (average of HR(i) weighted by read I/O rate)

PHR(i) - the projected read hit ratio of device i, computed below

APHR - average projected hit ratio (target provided by user)

If APHR >= AHR

$$\text{then} \quad PHR(i) = HR(i) + \frac{APHR - AHR}{1 - AHR}(1 - HR(i))$$

If APHR < AHR

$$\text{then} \quad PHR(i) = HR(i) - \frac{AHR - APHR}{AHR} HR(i)$$

Since the above algorithm increases high device hit ratios less than low device hit ratios when the average increases, the overall effect is to produce a set of hit ratios that approach the average. The function is simple and has been validated using a trace driven simulator (CAA) which provides hit ratios for different cache sizes.

4.3 CACHE SIZE PROJECTION

Projecting from one cache size to another is accomplished by giving the user the current cache size and average hit ratio for a control unit and allowing the specification of a different cache size. DCAT uses old cache size, new cache size, and old average hit ratio to first determine a new average hit ratio. The new average hit ratio is then used with the old hit ratios for each device to generate new hit ratios by the method described previously.

To determine a new average hit ratio for the new cache size, additional data is used by DCAT. This data is a table showing read hit ratio vs. cache size per MB of back store for each workload. This table has been generated from analysis of numerous measurements at computer installations using cached DASD.

The algorithm for mapping from one cache size to another is as follows:

1. Using the old average hit ratio and the function of read hit ratio vs. cache size per back store size, determine what the cache size per back store is for the old average hit ratio. Note that this value may or may not be identical to the actual cache size per back store.
2. Multiply the cache size per back store determined in step 1 by the ratio of the new cache size divided by the old cache size. Thus, if the new cache size is 48 MB and the old cache size was 32 MB, we would multiply the step 1 cache size per back store by 1.5.
3. Using the resulting cache size per back store from step 2 and the function of read hit ratio vs. cache size per back store size again, determine the new average read hit ratio.
4. Finally, generate the DASD hit ratios by the method described previously for projecting from one average hit ratio to another.

Clearly this method needs the function of average hit ratio vs. cache size per back store size to hold for all users with the same workload. Different workload types indeed have different characteristics for this average hit ratio function. For example, a workload such as TSO that is extremely favorable for caching quickly generates a high hit ratio at low values for cache size per back store size, then slowly builds from there. With the technique described above, and if a low old average hit

ratio is given for TSO, then almost any increase of cache size per back store size will show a dramatic increase in hit ratio. On the other hand, if the old average hit ratio is already high (.90 or larger), then it takes sizeable cache increases to get a significant increase in the average hit ratio.

The projection based on cache size change can be validated easily using the CAA tool that provides hit ratios for different cache sizes. The projected hit ratios are checked against the actual hit ratios. The method described above consistently provides hit ratios that are within an acceptable delta from the actual hit ratios.

5. VALIDATION

Validation of DCAT can be broken into two questions. The first question is: How accurate a set of workload parameters does DCAT generate during its calibration step? The second question is: How accurately does DCAT project to different environments or different workloads?

The calibration step consistently generates a set of workload parameters that provide model results for response and service times that are within 1% of the monitor values in MVS/XA environments. This may be comforting to the DCAT user when reported at the end of the calibration step; but in reality, generating a single set of workload parameters which matches the input values is easy. There are a multitude of sets of workload parameters which would satisfy the monitor data.

For example, when deciding how to divide the connect time between search and data transfer, we could assume no search with all data transfer, or all search with no data transfer, or somewhere in between. Unfortunately there is no monitor data available to provide hints. The split between search and data transfer is important when projecting to a cached control unit since the search time will be reduced but the data transfer time will not. DCAT divides the search and data transfer based on ratios measured from installations with similar workloads.

The result of all this is that the answer to the first validation question on accuracy of the calibration step is answered by looking at how well we are able to use this workload to project to different environments and workloads.

Many "what if" questions can be answered with the DCAT projection option as mentioned in Section 1. Each of the "what if" questions can be validated using various configuration and workload assumptions. We would like to present the results of three sets of validation runs here. The methodology involved obtaining measurements of base and target configurations and workloads which contained a device migration or I/O rate growth scenario. The base was entered in DCAT and a calibration step performed to generate a set of workload parameters. DCAT was then used to predict the performance of the target configuration or I/O rate. The model projected response times and service times were then compared to the measured data for the actual target benchmark. The workloads used were TSO and IMS benchmarks run in a laboratory environment. The configuration included one control unit, either 3880 model 3 or 3880 model 23 with 16MB to 32MB cache storage. The control unit had dedicated channel paths. The devices measured were: sixteen 3380 standard actuators, sixteen 3380D's, or eight 3380E's. Some runs involved accesses from multiple hosts. The first set of comparisons demonstrates how well DCAT projects growth in I/O rate. Table 1 lists eleven runs of workload growth with growth rates from 22% to 102%. The naming convention for run ID's are as follows: the first three characters, TSO or IMS, indicate the workload; the fourth, N or C, states 3880 model 3 or model 23 (cached); the last one is just a hexadecimal run number. The ratios of projected service times over measured service times are between 0.96 to 1.03. The similar ratios for response times are between 0.94 to 1.08.

TABLE 1. Validations for Workload Growth

RUN ID	BASE			PROJECTED			MEASURED		PROJ/MEAS	
	IO	ST	RT	IO	ST	RT	ST	RT	ST	RT
Single Host										
TSON1	102	21.0	24.6	142	24.9	32.8	24.6	32.6	1.01	1.01
TSON2	102	22.6	32.6	142	27.1	59.7	26.8	62.3	1.00	0.96
IMSN3	78	18.1	21.8	110	19.2	26.6	19.9	26.1	0.97	1.02
IMSN4	114	19.9	26.1	151	21.8	38.0	22.6	35.3	0.96	1.08
Multiple Host										
TSON5	135	18.3	25.9	165	20.0	33.9	19.7	33.7	1.02	1.01
TSON6	70	17.1	21.2	92	17.6	23.7	17.4	23.8	1.01	0.99
TSON7	92	17.4	23.8	125	18.4	30.1	18.1	30.6	1.02	0.98
TSON8	70	17.1	21.2	125	18.6	29.4	18.1	30.6	1.03	0.96
TSON9	55	18.3	20.8	111	20.9	29.0	21.1	28.2	0.99	1.03
TSONA	76	18.9	22.4	106	20.5	26.9	20.9	28.5	0.98	0.94
TSOCB	53	11.5	13.3	104	12.9	16.3	12.9	16.6	1.00	0.98

The second set of comparisons demonstrates how well DCAT projects performance when device migration is involved. There are seven pairs of runs for single host environments. Each pair consists of a 1-to-1 migration and a 2-to-1 migration with the same base. The 1-to-1 run represents the migration from sixteen standard 3380's to sixteen 3380D's. The 2-to-1 run has eight 3380E's as targets. These 14 runs are listed in Table 2. The naming convention is similar to that used in Table 1. The ratios of projection over measurement are 0.97 to 1.06 for service times and 0.94 to 1.16 for response times.

TABLE 2. Validations for Device Migration

RUN ID	BASE		PROJECTED		MEASURED		PROJ/MEAS	
	ST	RT	ST	RT	ST	RT	ST	RT
1-to-1 Mapping								
STSON1	23.4	28.7	21.0	25.1	21.0	24.6	1.00	1.02
STSON2	28.1	39.7	23.9	31.9	24.6	32.6	0.97	0.98
SIMSN3	18.1	21.8	16.7	19.8	16.8	20.0	1.01	0.99
SIMSN4	19.9	26.1	17.8	22.5	17.9	23.0	1.00	0.98
SIMSN5	22.6	35.3	19.7	27.5	19.6	28.0	1.00	0.98
STSOC6	12.3	16.1	11.6	14.4	11.6	15.4	1.00	0.94
STSOC7	14.5	22.1	13.9	20.2	13.1	19.0	1.06	1.06
2-to-1 Mapping								
STSON8	23.4	28.7	23.7	37.0	22.6	32.6	1.05	1.13
STSON9	28.1	39.7	25.7	51.9	26.5	52.7	0.97	0.98
SIMSNA	18.1	21.8	18.0	22.5	17.2	22.0	1.05	1.02
SIMSNB	19.9	26.1	19.0	25.8	18.4	27.0	1.03	0.96
SIMSNC	22.6	35.3	20.8	35.7	20.1	38.0	1.03	0.94
STSOCD	12.3	16.1	12.4	20.9	12.2	18.0	1.03	1.16
STSOCE	14.5	22.1	13.5	28.2	13.6	24.4	0.99	1.16

The third set of comparisons demonstrates how well DCAT projects when a control unit is upgraded with cache storage. Essentially, devices are moved from a uncached 3880 model 3 to a cached 3880 model 23. There are 12 cases listed in Table 3. The fifth character "C" of the run ID indicates that this is a cache projection. The ratios for service times are between 0.93 to 1.11, and the ratios for response times are between 0.88 to 1.12. Prior to any validation work, we have generally agreed that 20% is the acceptable tolerance for response time and physical service time differences.

TABLE 3. Validations for Cache Migration

RUN ID	BASE		PROJECTED		MEASURED		PROJ/MEAS	
	ST	RT	ST	RT	ST	RT	ST	RT
IMS Workload								
SIMSC1	18.1	21.8	9.5	11.4	9.3	12.0	1.02	0.95
SIMSC2	19.9	26.1	10.6	13.4	10.2	14.0	1.04	0.96
SIMSC3	22.6	35.3	12.2	16.8	11.2	17.0	1.09	0.99
SIMSC4	16.8	20.0	9.0	10.4	8.7	10.0	1.03	1.04
SIMSC5	17.9	23.0	9.6	11.8	9.0	11.0	1.07	1.08
SIMSC6	19.6	28.0	10.9	14.5	9.8	13.0	1.11	1.12
SIMSC7	17.2	22.0	9.0	11.1	8.7	11.0	1.03	1.01
SIMSC8	18.4	27.0	9.8	13.1	9.2	13.0	1.07	1.01
TSO Workload								
STSOC9	23.4	28.7	11.5	14.1	12.3	16.1	0.93	0.88
STSOCA	21.1	28.2	11.9	15.9	12.8	16.9	0.93	0.94
MTSOCB	18.3	20.8	10.2	11.5	11.5	13.2	0.89	0.87
MTSOCC	28.1	39.7	13.9	19.8	14.5	22.1	0.96	0.90

6. SUMMARY

The normal use of DCAT involves collecting monitor data and running it through the calibration option. The resulting workload is then used to project to different environments and workloads.

The crucial algorithms for the subsystem model and projection -- namely, the RPS miss delay calculation, the seek time projection, and the cache migration -- have been in a process of change and evolution as we obtain more measurement data for validation. The algorithms were developed and validated with the help of special purpose simulation models as well as the measurements described. The validation efforts done to date show that the projection algorithms are consistently providing results within an acceptable range.

DCAT has been getting worldwide usage within IBM since 1985, and the user response has been favorable. In the US alone, the monthly averages for DCAT indicate 300-400 users accounting for 1500-2000 sessions. DCAT has helped IBM system engineers working with customers to suggest cost-effective configurations which meet their requirements.

ACKNOWLEDGEMENTS

DCAT has been successful only because of the expertise and efforts of many people. A few of them are: Linda Duyanovich for her pioneer work on DCAT and many helpful suggestions for DCAT and this paper; Gordon Nouhan, Annette Spaeth, and Ruth Cintron for their development efforts to mold the above ideas into useful products; Ed Nauss for his efforts in laying a good foundation for the subsystem model with his simulation; Marc Goldfeder whose ideas make up the cache modeling and the cache projection algorithms; and Bruce McNutt for his work on seek projections upon which DCAT's technique was built.

REFERENCES

1. Friesenborg S. and Wicks R., "DASD Expectations", GG22-9363-02 (1985), available through IBM branch office.
2. Kleinrock L., Queueing Systems, Volume 1: Theory, John Wiley & Sons, New York (1975).
3. McNutt B., "The Impact of 2-to-1 Volume Folding on Seek Time: A Comparison of Five Cases", CMG Proceedings, pp.427--433 (1985).

A MULTIPLE WORKLOAD APPROACH TO CACHE PLANNING

Bruce McNutt and James W. Murray
International Business Machines Corporation
San Jose, California

ABSTRACT

Cached controllers can provide a powerful and cost-effective tool for speeding up the I/O activity at an installation. This is true even if the cache is simply installed and turned on. Provided that a large enough amount of cache memory is provided, no particular planning is required to obtain good results.

Long-term planning can, however, enhance the cost-effectiveness of cache and can simplify the cost-justification process, by taking into account the value of high performance for specific applications. The purpose of this paper is to introduce a set of techniques which may be helpful in choosing a strategy for the use of cache which best meets the needs of a given installation, if a large number of possible options is being considered. This is done by breaking down the data to be cached into workload-related volume groups, or "pools". For example, an installation might be considering the purchase of either one, two, or three cached controllers, which would be used to cache some subset of the key system data, the TSO user data, and three data bases of various types. For each cache, a choice among several memory sizes must also be made. An approach to this problem is introduced in which the locality of reference is estimated separately for each individual pool which may share the use of a cached controller. Requirements for cache memory are also estimated for each pool. The proposed approach then allows these estimates to be applied to shared caches containing any combination of the original pools. This procedure makes possible the efficient examination of a large number of potential cached configurations.

1. INTRODUCTION

Cached controllers can provide a powerful and cost-effective tool for speeding up the I/O activity at an installation. This is true even if the cache is simply installed and turned on. Provided that a large enough amount of cache memory is provided, no particular planning is required to obtain good results. If there is a question as to the amount of memory which is required, then this can be resolved through the use of a cache simulation tool such as Cache Analysis Aid [1, 2].

Long-term planning can, however, enhance the cost-effectiveness of cache and can simplify the cost-justification process, by taking into account the value of high performance for specific applications. The purpose of this paper is to introduce a set of techniques which may be helpful in choosing a strategy for the use of cache which best meets the needs of a given installation, if a large number of possible options is being considered. This is done by breaking down the data to be cached into workload-related volume groups, or "pools".

As an example, imagine an MVS shop where we have put together the following list of volume groups ("pools") as caching candidates:

1. A select group of performance critical ("key") system volumes, containing master catalogs, heavily used program libraries, the IMS long and short message queues, the Hierarchical Storage Manager (HSM) and Resource Access Control Facility (RACF) control data sets, and other control and/or widely shared data.
2. A decision support system data base, used for interactive planning and analysis by upper level management.
3. Twenty TSO user volumes.
4. An inventory data base, used for tracking the status of warehouse items.
5. An accounts payable data base, used for batch billing and on-line inquiries.

Each of these candidate volume pools accounts for a different proportion of the total I/O activity. In addition, each has a certain service priority; the key system volumes must have the highest priority of good response, the priority of the decision support system comes second, and the remaining pools are all to be treated as equal. Finally, each of the candidate pools will benefit differently from the use of cache.

We are now confronted with the problem of selecting which subset of these candidate pools to place under a cache, how many caches to buy, how to mix the cached pools onto the available caches, and how much cache memory will be needed. For example, we might choose to place the key system volumes on cache 1 and the decision support system data base on cache 2; or to combine the decision support system with the inventory data base on cache 2 while cache 1 continues to handle the key system data; or to place both the key system volumes as well as the decision support system on cache 1, while cache 2 contains the TSO user volumes; and so on.

This paper shows that the time-honored approach of problem decomposition can be applied in addressing questions of the kind described above. Through the techniques developed below, individual data pools can be examined first, and the performance of collections of pools can be projected by building upon the pool-by-pool results.

The data used to drive the proposed approach is obtained by applying a cache simulation, such as that performed by the Cache Analysis Aid, to a Generalized Trace Facility (GTF) trace of I/O activity, including channel control words (CCW's). Cache simulations can be used to estimate the hit ratio and read/write ratio of the cache assuming that the backing store contains any specified group of volumes (e.g. any data pool), and assuming a range of cache memory sizes which might be selected [1 - 3]. These parameters are the key to determining the performance benefits which the cache is likely to achieve [1, 4 - 5]

In the multiple-workload approach, information obtained from cache simulations is used to describe the behavior of the cache individually for each pool. This paper introduces a set of techniques which allows the pool-by-pool simulation results to be combined to obtain a prediction of the overall performance of a cache which contains two or more pools. To estimate the impact on such a cache of adding a pool, removing a pool, or caching some other combination of pools, the same basic performance data for each pool can be re-used, as many times as needed, in different combinations.

The key making the above strategy of decomposition work is to view the hit ratios obtained from the simulation as being a function of the average time which a track spends in the cache. The construction of this view of the data is straightforward if the data provided from the simulation includes a broad range of hit ratios and corresponding cache sizes. The central result reported in this paper is that, once the hit ratio of each pool is expressed as a function of the average time in cache, it is possible to predict with very high accuracy the hit ratio on a cache containing mixed pools, as a function of either the average time in cache or of cache memory size.

These results mean that in the example MVS installation discussed above, a very broad range of possible configuration options can be explored by using the results of only five cache simulation runs (one for each pool). Once these five runs have been performed, planning can proceed by mixing and matching the five pools onto various numbers of caches. For a given cache containing a specific mixture of pools, the hit ratios and read/write ratios for each pool are estimated separately, as are the corresponding cache memory requirements. Since these data are available separately for each pool, the

actual benefits can be assessed easily for each pool depending on the relative priority of service for that pool. The total cache memory requirements for the cache as a whole are obtained simply by adding the requirements for each pool. The overall hit ratios and read/write ratios can also be obtained by aggregating the individual pool data.

The strongest results of this paper are a number of test cases, based on live installation data, where the above procedure for combining pool data was employed successfully to predict the performance of a mixed cache configuration. These results represent a variety of different computing environments, but may not necessarily be representative of all environments and configurations. For this reason, the paper starts off in Section 2 with an overview of the trace data, collected from twelve different installations and subdivided into five standard pools, which was used as a basis for the test cases. Section 3 then motivates the idea of expressing the hit ratio as a function of the average time in the cache, and explains why we should expect it to be possible to combine a function of this kind across pools. Section 3 also develops the specific procedure used to obtain this function from simulation results, and proposes the method which would be used to combine two or more such functions in order to evaluate the case of a mixed cache.

Section 4 tries out the idea of hit ratios expressed as a function of the average time in cache. For each standard pool (CICS data base, IMS data base, TSO user data, BATCH data, and KEY SYS data), and for each installation where a pool of this kind was traced, we present the hit ratio curve as a function of the average time in the cache. These curves provide an interesting comparison of the differences and similarities in the cache performance of a given type of data across a number of installations, and they yield some useful clues about the range of performance which may be achievable by using cache for specific types of installation data. Section 5 then tests the major premise of the paper. For a selected list of pool pairs (not all possible combinations!), predictions are developed of the hit ratio as a function of the average time in the cache. These predictions are tested against the result of a simulation based on merged traces of the activity from both pools. The results show that the predicted hit ratio is never off by more than 1.4 percentage points.

2. OVERVIEW OF THE DATA

	DATA POOL									
	IMS		TSO		CICS		BATCH		KEY SYS	
	NO. OF DEV	TOT I/O RATE	NO. OF DEV	TOT I/O RATE	NO. OF DEV	TOT I/O RATE	NO. OF DEV	TOT I/O RATE	NO. OF DEV	TOT I/O RATE
COMPLEX										
A			84	92.2	3	3.2			39	132.7
B	55	72.9	48	37.7	5	1.1			26	153.3
C	73	201.9	45	13.7	55	45.0			36	211.3
D	56	132.9	14	34.7			50	67.7	23	80.8
E			133	207.8					27	163.6
F	46	199.1	67	132.5			95	120.3	16	79.8
G	84	404.2			5	8.5	74	39.1	23	65.3
H	34	72.2	21	23.4			20	31.0	15	25.8
I					456	631.1	155	39.7	42	133.6
J					347	424.3	269	102.7	34	87.5
K					329	175.7	226	30.1	40	145.9
L	110	117.4	55	23.5			47	9.3	34	80.4

Figure 1. Overview of Trace Data

Figure 1 presents an overview of the data which we will be talking about in the rest of the paper. This data is an outgrowth of an ongoing project to collect information about how DASD are used in live installation environments.

As Figure 1 shows, trace data was collected from twelve installations. At each installation, data was collected over a period of at least 15 minutes (for complexes A and D the period was thirty minutes). One purpose of the present study was to investigate the relationship between the effectiveness of the cache and the general type of data contained in the backing store. The volumes being traced were therefore divided into five standard pools according to the general category of data contained on the volume:

- IMS: IMS data base storage.
- CICS: CICS data base storage.
- TSO: TSO user data (in some cases administered by HSM)
- BATCH: Batch application data
- KEY SYS: Control and/or widely shared data central to system operation, including master catalogues, RESLIB data, heavily used program libraries, the HSM and RACF control data sets, the JES2 checkpoint data set, the IMS long and short message queues, and so forth. This category does *not*, however, include page data sets, scratch data sets, spooled data, or IMS logs.

For each category of data at each installation, Figure 1 presents the number of volumes on which data was collected and the average I/O rate of these volumes.

Note that some pools of data contained a very large number of devices. This is a result of the set of standard data pools which were adopted for the purpose of the study. At a given installation, it would have been possible to subdivide any of the various data base or system pools into smaller, more installation-dependent categories. Smaller and more manageable pools would be essential if we were to attempt to do detailed planning for the use of cache at some some particular installation presented in Figure 1. The appropriate partitioning of the volumes of each of the twelve installations into small, installation-dependent categories, however, is beyond the scope of what is being attempted in the present study.

As a result of the use of the standard pool categories shown in Figure 1, we must therefore introduce certain conventions to deal with the possibility of very large pools:

CONVENTION 1: Normal constraints on the maximum number of volumes which may be placed under a cache and/or the maximum cache memory size are not explicitly modeled. Instead, we will compute how much cache memory would be required to handle the activity on each pool of volumes, without regard to size, as though all of this activity were going to a single large cache. (In practice, of course, these volumes would be dealt with in much smaller, application-specific groups).

CONVENTION 2: The computation of the hit ratio is not adjusted to remove write activity from the numerator and denominator. This is due to the problems in obtaining GTF/CCW traces for large numbers of devices. In simulations based on traces of CCW data, the CCW's associated with each I/O are examined to identify those I/O's which cause a write operation to occur. It is then possible to adjust the hit ratio so as to exclude these write I/O's from both the numerator (hits) and the denominator (I/O's to the cache). In a separate preliminary study of this adjustment, however, little if any quantitative difference was found between the adjusted, or "read", hit ratio and the unadjusted hit ratio computed over all I/O's. As a result, the decision was made to perform the present study using SIO traces rather than the full CCW traces. Collection of SIO trace data for all volumes at an installation is efficient and simple; potential problems which might interfere with the

collection of CCW data for so many volumes, such as tape overrun or large amounts of CPU overhead, are avoided. Since CCW data is not included in the trace, however, the unadjusted value of the hit ratio must be used.

Most of the pools presented in Figure 1 contained 3380 devices exclusively. However, some pools (particularly at complexes I, J, and K) contained both 3380 and 3350 devices. To simulate the cache activity of data moved under a cache from a 3350 device, the usual assumption of 2-to-1 mapping was made:

CONVENTION 3: Each pair of adjacent tracks on a 3350 device is assumed to correspond to a single track for the purpose of evaluating cache activity. More specifically, 3350 tracks 0 and 1 are assumed to form one cached track, 3350 tracks 2 and 3 are assumed to form the next cached track, and so forth.

3. THE ROLE OF CACHE RESIDENCY TIME

This section develops in a more concrete and specific way the proposed scheme for projecting the performance of a cache. The starting point for this scheme is the observation that ordinary cache simulation results, which project hit ratios for a range of cache memory sizes, contain hidden information which is worth bringing to the surface. As the memory size is varied over the range adopted for the simulation, there is also an implicit variation in the average residency time spent by a track in the cache. This is what actually causes the hit ratio to change -- the longer a given track stays in the cache, the larger is the probability that it can be reused during its visit.

The quantitative relationship between the cache size and the average time-in-cache can be specified using Little's law, as described in Subsection 3.1. But for the moment, it is worth while to focus on an important qualitative difference between these two parameters: namely, *the relationship between hit ratio and time-in-cache makes sense for every individual track*. We can talk about one single track in isolation from the others. As an extension of this, we can also talk about any specific group of tracks (such as a specific pool of data) in isolation from any other group of tracks. This is impossible when we think of the hit ratio as a function purely of cache size -- the cache size is defined only with respect to the entire cache.

Knowing the average time-in-cache for the cache as a whole does not, of course, mean that we know the average time-in-cache for each individual track; all tracks do not necessarily have the same average residency time. Similarly, all pools which share a cache do not necessarily have the same average residency time. But this caveat is not as damaging as it may seem at first glance.

Suppose, for example, that two pools share a cache. The overall track residency time for either of the two pools is an average of the single-reference residency time (the average residency time for tracks that are referenced exactly once during a given visit) and the multiple-reference residency time (the average residency time for tracks that are referenced more than once). Now, the single-reference residency time is just the average time it takes for a track to migrate from the top to the bottom of the LRU list, assuming that the track is never referenced after appearing at the top of the list. Since tracks from either pool must traverse the same LRU list, both pools must have the same single-reference residency time to a very close approximation. In addition, the residency time for tracks making multiple-reference visits will typically be close to the single-reference time, since the references to these tracks typically occur in a burst. Thus, it is natural to hypothesize that both pools will have average residency times which are, if not equal, at least very close to each other.

Moreover, we shall see in Section 4 that the hit ratio changes only gradually with the time-in-cache. In connection with the reasoning above, this suggests that it may be safe to ignore variations in the individual cache residency times of pools sharing a cache, insofar as they affect the pool hit ratios. Knowledge of the average cache residency time for the cache as a whole may be sufficient to predict the behavior of each individual pool. If we pull together the entire line of reasoning given above, then, we get the following

HYPOTHESIS: Whether a pool is by itself on a cache or shares the cache, the hit ratio experienced by the pool can be predicted as a function of the average cache residency time for the cache as a whole. Except for this relationship, the pool can be treated as being independent of any other pools served by the cache.

This is the hypothesis which the remainder of the paper will pursue. Rather than portraying hit ratio as a function of cache size, we will try portraying it as a function of the average time-in-cache. This information will be developed for each pool for which trace data is available. According to the hypothesis, we should then be able to project the hit ratio as a function of the time-in-cache for a cached controller containing a mixture of two or more pools. This is done merely by projecting the hit ratio for each pool, treated independently, and then aggregating the pool hit ratios to obtain the combined hit ratio for the cache as a whole.

Before diving into details of the study, it is interesting to note that a family resemblance exists between the line of reasoning above and the concept of "working sets" used in the implementation of demand paging [6]. In both cases, the idea is to try to identify how much of a shared memory resource is needed by each individual user of the resource. The strategies necessary in the two situations are rather different, however, since in the case of managing the cache there is no dynamic control over who may use the cache at any given time. Also, there is no mechanism for imposing a specific time window over which the working set is defined. Thus, the issue of consistency of residency times across pools arises, which does not seem to have a direct analogue in the analysis of working sets.

3.1 How it Works

This section describes the specific equations used to implement the game plan described above. The first order of business, in Subsection 3.1.1, will be to lay out the specific relationships which allow us to translate back and forth between cache size and cache residency time. Subsection 3.1.2 then provides the detailed proposal for how we can project the performance of a cache containing a mixture of pools.

3.1.1 Calculating the Cache Residency Time

The first step in any method of predicting the performance of a cache, including the one being proposed, is to collect a GTF/CCW trace of the I/O activity which the cache will support. A simulation tool must then be run against this trace to determine, for a range of possible cache memory sizes, the hit ratio and read/write ratio achievable by the cache for this type of I/O activity.

In the technique being proposed, simulation results of this kind are obtained for each pool i, $1 \le i \le P$, where P is the number of pools. The output of the simulation for each pool contains the projected hit ratios and read/write ratios for a range of possible cache memory sizes. This yields a series of cases, one for each cache memory size, that we can talk about in terms of the following variables. To emphasize that the simulation results are obtained separately for each pool i, each variable is subscripted by the pool number:

M_i = The size of the cache, measured in megabytes.

S_i = The size of the cache, measured in track slots. S_i can be approximated as being a multiple of the cache megabytes. This is not strictly true, due to granularity effects, but it is close enough. A good (slightly pessimistic) rule is that there are twenty-one 3380 (or equivalent) track slots per cache megabyte.

T_i = The average track residency time of the cache

r_i = The I/O rate to pool i during the trace period being examined.

h_i = The hit ratio of pool i during the trace period, adjusted to remove writes from the numerator and denominator. This is the hit ratio usually reported by cache simulation tools.

H_i = The hit ratio of pool i during the trace period, not adjusted to remove writes.

C_i = The read/write ratio of pool i during the trace period, providing that there are writes. If there are no writes, then C_i is not defined.

w_i = The fraction of I/O's to pool i, during the trace period, which are writes. If there are no writes then $w_i = 0$. Otherwise, $w_i = (1 + C_i)^{-1}$

Using these variables, we can now investigate how the cache size relates to the cache residency time T_i. This is done by viewing the LRU list as a queuing system; tracks enter the list, stay for an average period of time T_i, then leave again. Both the rate of entry into the system (the number of read misses per second) and the population of the system (fixed at S_i) are known. Therefore, by Little's law, we have

$$\begin{pmatrix}\text{popu-}\\\text{lation}\end{pmatrix} = \begin{pmatrix}\text{resi-}\\\text{dency}\\\text{time}\end{pmatrix} \times \begin{pmatrix}\text{arrival}\\\text{rate}\end{pmatrix} \tag{1}$$

$$S_i = T_i \times \left[\begin{pmatrix}\text{read miss}\\\text{probability}\end{pmatrix} \times \begin{pmatrix}\text{I/O}\\\text{rate}\end{pmatrix} \right]$$

$$S_i = T_i (1 - h_i)(1 - w_i) r_i$$

or equivalently,

$$T_i = \frac{S_i}{(1 - h_i)(1 - w_i) r_i} \tag{2}$$

For example, suppose that we have just collected a GTF/CCW trace of the TSO user pool at the installation described earlier. The trace covers a period of 15 minutes, and during this period of time the total I/O rate to the pool was 70 I/O's per second. We run a cache simulation against the trace data, and find that during the simulation period the read/write ratio was 9, and that with a cache size of 16 megabytes we can obtain a hit ratio of 85 percent. Now, the cache size of 16 megabytes corresponds to approximately S_{TSO} = 16*21 = 336 track slots; and the fraction of I/O's which are writes is $w_{TSO} = (1 + 9)^{-1}$ = 10 percent. Therefore, we calculate from (2) that

$$T_{TSO} = \frac{336}{(1 - 0.85) \times (1 - 0.1) \times 70} = 35.6 \text{ seconds}$$

If this reasoning is repeated for a range of cache sizes, then a chart of hit ratio as a function of time in the cache is obtained for the TSO pool. This is the technique used in Section 4 to develop the charts presented by Figures 2 - 5. These charts show hit ratio as a function of time in cache for each pool included in the study.

In the case of the pools traced in the present study, the values of w_i are not known accurately due to the use of SIO, rather than CCW, traces. It *is* known, however, that the values of w_i are small, since these pools do not contain any scratch, spool, paging, or IMS log data. As a rough approximation, we set $w_i = 0$ in (1) and (2), and use H_i in place of h_i. This results in a small error in computing the time-in-cache corresponding to a given hit ratio. Since it affects a given pool in the same way whether the pool is by itself or on a mixed cache, however, a small error of this kind does not interfere with our ability to test the hypothesis of the paper. As shown in Section 4, hit ratio estimates are insensitive to this form of error in any case; the hit ratio changes only gradually with the time in cache.

3.1.2 Estimating the Performance of a Mixed Cache

Once a chart is available for each pool showing hit ratio as a function of time-in-cache, we are ready to develop a chart for the case of any desired combination of pools. The idea is to combine the charts for several pools sharing a cache through ordinary averaging. Since the hit ratio is defined as a fraction of read I/O's, the average must be weighted by the read I/O rate.

More formally, suppose that the quantities r_i, w_i, and C_i, and the functions h_i and S_i of T_i, have been obtained for each pool i, $1 \le i \le P$. Given a cache in which these pools are combined, we are now able to compute the aggregate I/O rate $r_\bullet$, the aggregate fraction of writes $w_\bullet$, and the aggregate read/write ratio $C_\bullet$. In addition, we are able to compute estimates $h_\bullet$ and $S_\bullet$ of the aggregate hit ratio and aggregate cache memory size as functions of the overall cache residency time $T_\bullet$.

The combined I/O rate, of course, is

$$r_\bullet = \sum_{i=1}^{P} r_i \tag{3}$$

Since mixing of the pools has no effect on each pool's pattern of reads and writes, it is also immediate that

$$w_\bullet = \frac{1}{r_\bullet} \sum_{i=1}^{P} r_i w_i \tag{4}$$

and, assuming that there are writes,

$$C_\bullet = \frac{1 - w_\bullet}{w_\bullet} \tag{5}$$

Finally, we must develop a chart of hit ratio versus time-in-cache for the combined pools. The hypothesis of independent pool behavior is used to estimate the individual hit ratio and corresponding memory requirements of each pool within the combined cache:

$\hat{h}_i$ = the value of h_i which corresponds to a time-in-cache equal to $T_\bullet$, as specified in the the chart of hit ratio data previously obtained for pool i by itself.

$\hat{S}_i$ = $T_\bullet (1 - \hat{h}_i)(1 - w_i) r_i$ = the memory requirement of pool i obtained by plugging this hit ratio into (1).

The aggregate hit ratio is then obtained as a weighted average of the individual pool estimates:

$$\hat{h}_\bullet = \frac{1}{r_\bullet (1 - w_\bullet)} \sum_{i=1}^{P} r_i (1 - w_i) \hat{h}_i \tag{6}$$

If the quantities $\hat{h}_i$ were replaced by measured hit ratios on the combined cache, then (6) would yield the exact value of the aggregate hit ratio. The major point of the paper, however, is that these measurements on the combined cache are *not needed*. The simulations of each pool by itself provide a sufficient amount of data to produce sound estimates.

The total memory requirements of the combined cache can be estimated either as the sum of the estimated requirements for each pool, or by another application of Little's law:

$$\hat{S}_\bullet = \sum_{i=1}^{P} \hat{S}_i \tag{7}$$

$$= T_\bullet (1 - \hat{h}_\bullet)(1 - w_\bullet) r_\bullet$$

It can be shown that these two methods of calculation are equivalent.

Example:

Consider a cache in which the TSO pool described earlier is combined with the KEY SYS pool. By analyzing each pool individually as described in Subsection 3.1.1, we estimate that that if the cache residency time is 30 seconds, then the TSO and KEY SYS pools can achieve hit ratios of 84 and 93 percent respectively. The I/O rates of the two pools during the trace period were 70 and 50 I/O's per second respectively, and the read/write ratios were 9 and 6 respectively. Corresponding to the read/write ratios of 9 and 6, the fractions of write activity for the two pools are $(1 + 9)^{-1}$ = 0.100 and $(1 + 6)^{-1}$ = 0.143 respectively.

We begin the analysis of the planned cache by computing the aggregate I/O rate and read/write ratio according to equations (3) - (5):

$$r_* = 70 + 50 = 120$$

$$w_* = \frac{1}{120}(70 \times 0.100 + 50 \times 0.143) = 0.118$$

$$C_* = \frac{1 - 0.118}{0.118} = 7.47$$

Based on the data collected for each pool by itself, we estimate that for a mixed cache with a residency time of $T_* = 30$ seconds, we can obtain hit ratios of

$$\hat{h}_{TSO} = 0.84$$

$$\hat{h}_{KEY\,SYS} = 0.93$$

By (1), the memory requirements corresponding to these hit ratios are

$$\hat{S}_{TSO} = 30 \times (1 - 0.84) \times (1 - 0.100) \times 70 = 302 \text{ track slots}$$

$$\hat{S}_{KEY\,SYS} = 30 \times (1 - 0.93) \times (1 - 0.143) \times 50 = 90 \text{ track slots}$$

Using these hit ratios and storage requirements for each component of the activity, we can now use (6) and (7) to estimate the hit ratios and storage requirements overall:

$$\hat{h}_* = \frac{70 \times (1 - 0.100) \times 0.84 + 50 \times (1 - 0.143) \times 0.93}{120 \times (1 - 0.118)} = 0.877$$

$$\hat{S}_* = 302 + 90 = 392 \text{ track slots} = 18.7 \text{ MB}$$

Since a cache controller cannot be configured to have a size of precisely 18.7 MB, we must choose either the next size down or the next size up. The required memory size is probably close enough to 16 MB so that good performance can still be achieved by a 16 MB cache -- but this would not leave any room for future growth in the workload. Alternatively, we might choose a 32 MB cache. In this case the cache performance will exceed requirements (not necessarily a bad outcome), since the average time-in-cache will be greater than the required 30 seconds. Another idea might be to try caching an additional small pool of data, such as the decision support system. The resulting combination of three pools might take greater advantage of a 32 MB cache.

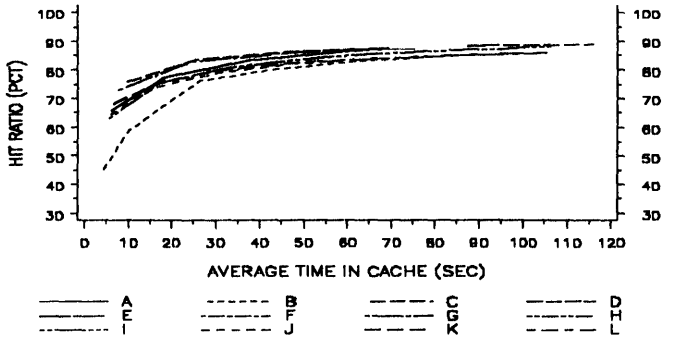


Figure 2. Hit Ratios for CICS Pools

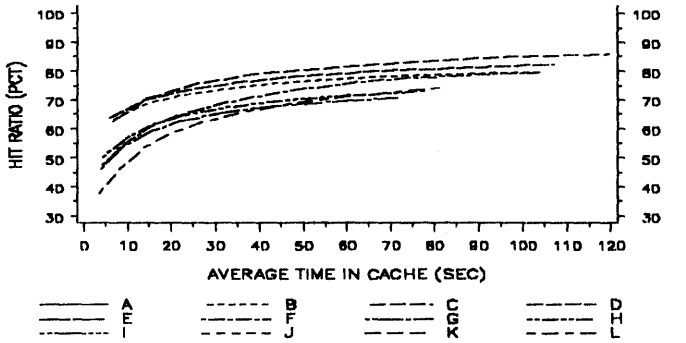


Figure 3. Hit Ratios for IMS Pools

4. A NEW VIEW OF HIT RATIO DATA

The next step is to try out the idea of plotting hit ratio as a function of time-in-cache. Figures 2 - 6 present the results of the analysis described in Subsection 3.1.1 as applied to the CICS, IMS, TSO, BATCH, and KEY SYS pools respectively. The data for each pool observed at a given installation is shown by a specific curve, which is identified by the letter for the installation as defined in Figure 1.

These hit ratio curves exhibit a number of interesting features. Note particularly that each curve seems to have a "knee" at or near a cache residency time of 20 seconds. An effect of diminishing returns begins to set in by the time the curve reaches 30 seconds, and there seems to be relatively little benefit associated with increasing the time-in-cache beyond 60 seconds. (An exception to this pattern may be the BATCH pools, which seem to show relatively less responsiveness to an increase in the cache residency time compared to the other pools).

Another important feature of the hit ratio curves is the degree of consistency which seems to exist for data of the same type across various installations. This is particularly true toward the right-hand side of each chart. At a time-in-cache of 60 seconds, it seems that we should be able to expect, with reasonably high confidence, that a CICS pool can achieve a hit ratio of 82-87 percent; an IMS pool can achieve a hit ratio of 70-82 percent; a TSO pool can achieve a hit ratio of 84-93 percent; and a KEY SYS pool can achieve a hit ratio of 84-94 percent. The performance of BATCH data seems more uncertain; this is probably dependent, among other things, upon the blocking of the portion of the BATCH data which is sequential.

Finally, it should be noted that in the region past about 20 seconds, the hit ratio is a gradual, rather than a highly sensitive, function of the cache residency time. This bodes well for the hypothesis offered earlier in the paper. If the average residency times for two pools sharing a cache are slightly different, and if we ignore the difference between the two, this would cause only a very small error in the resulting estimates of the pool hit ratios.

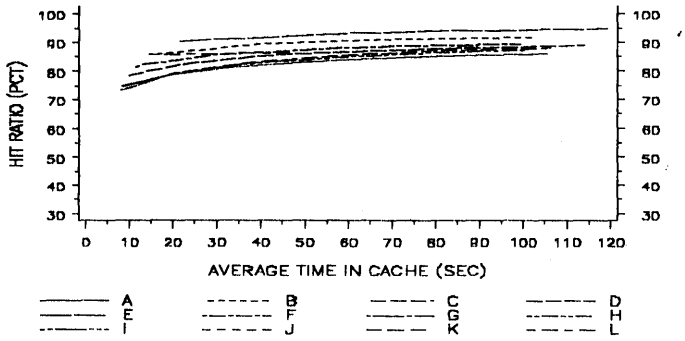


Figure 4. Hit Ratios for TSO Pools

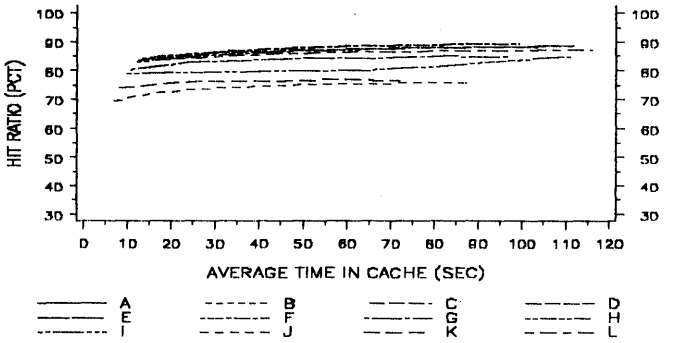


Figure 5. Hit Ratios for BATCH Pools

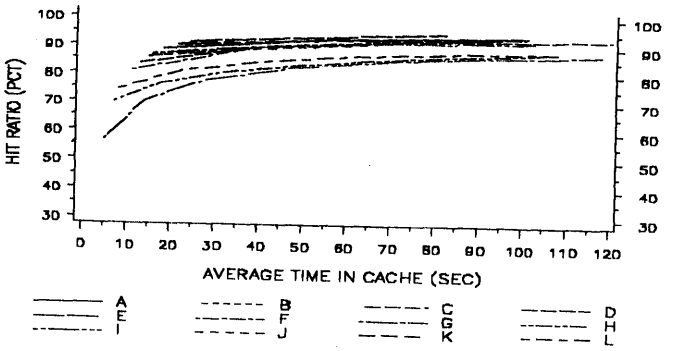


Figure 6. Hit Ratios for KEY SYS Pools

5. TEST RESULTS

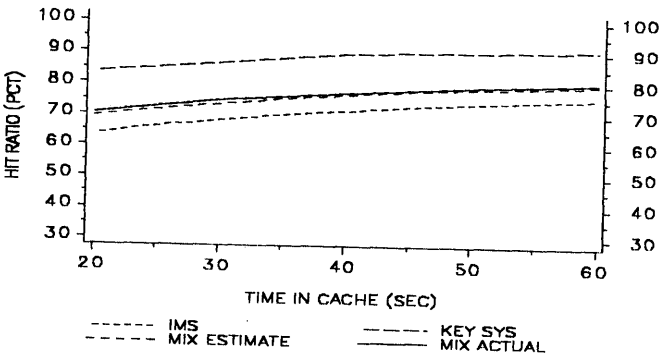


Figure 7. Predicted and Actual Mixed Pool Hit Ratio: KEY SYS and IMS Pools at Complex F

With the data of Figures 2 - 6 in hand, we are now able to test the hypothesis of Section 3. The approach to testing is based on (6) -- i.e., we will estimate the hit ratio of a mixed cache as though the hit ratio of each pool were an independent function of the cache residency time T_c. We will then see if this estimate is accurate enough so that (6) holds.

Note that for the purpose of testing the hypothesis of independent pool behavior, the definition of a pool is arbitrary. Two pools A and B can be treated as a single combined pool A + B if we wish. There is no physical distinction between these two methods of grouping the same data. Thus, it suffices to test the hypothesis for groups of two pools. The hypothesis can then be applied to larger numbers of pools simply be redefining the pool groupings.

The method of testing is illustrated in Figure 7. The top and bottom curves of the figure show the relationship of hit ratio versus time-in-cache for the

KEY SYS and IMS pools respectively at complex F. The dashed curve in the center shows the predicted hit ratio if these two pools were combined on the same cache, derived using (6). The solid line shows the actual hit ratio which was obtained by merging the I/O traces for the two pools and running a cache simulation against the merged trace. As the figure makes clear, very little error is introduced through the use of (6).

	MIXED HIT RATIO (PCT) FOR T =						
	20		30		60		
	ESTI-MATED	ERROR	ESTI-MATED	ERROR	ESTI-MATED	ERROR	MAX ERROR
COMPLEX; POOLS MIXED							
A; TSO, KEY SYS	84.7	-0.3	86.0	-0.8	88.9	-0.2	-0.8
B; IMS, KEY SYS	81.6	-0.4	83.4	-0.5	86.4	-0.2	-0.5
C; IMS, KEY SYS	81.9	-0.2	84.0	-0.4	87.4	0.0	-0.4
D; IMS, KEY SYS	77.1	-0.8	79.6	-0.7	83.7	-0.3	-0.8
E; TSO, KEY SYS	90.3	-0.0	91.3	-0.5	93.3	-0.5	-0.9
F; IMS, KEY SYS	69.3	-1.2	73.4	-1.2	80.1	-0.7	-1.2
F; IMS, BATCH	71.7	-0.3	75.1	-0.3	80.5	-0.3	-0.4
F; TSO, BATCH	84.3	-0.7	86.1	-0.1	88.2	-0.1	-0.7
G; BATCH, KEY SYS	76.5	-1.4	79.9	-0.7	83.9	-0.6	-1.4
H; TSO, BATCH	81.9	-0.4	83.6	-0.5	86.3	-0.3	-0.5
H; TSO, KEY SYS	78.0	-0.3	80.5	-0.3	84.6	-0.3	-0.3
H; IMS, BATCH	69.5	-0.3	72.4	-0.2	76.3	-0.2	-0.3
I; CICS, BATCH	76.4	-0.9	79.8	-1.0	84.6	-0.7	-1.0
J; CICS, BATCH	74.7	-0.8	78.0	-0.3	81.8	-0.4	-0.8
K; CICS, KEY SYS	82.7	-0.4	84.6	-0.3	87.4	-0.1	-0.4
L; IMS, KEY SYS	66.0	-1.2	70.1	-0.9	76.6	-0.5	-1.2

Figure 8. Tests of the Pool Independence Hypothesis

Figure 8 presents all 16 specific pairs of pools which were selected for the purpose of testing the pool independence hypothesis. The criteria used in constructing these pairs were,

1. Contrasting types of activity, and
2. Reasonable balance in the I/O rates (so that each pool would make some contribution toward the average hit ratio).

The figure shows the predicted hit ratio for each pair of pools at residency times of 20, 30, and 60 seconds, obtained from (6). For each pair of pools, a cache simulation was run against the merged trace. The error following each predicted hit ratio in Figure 8 is the difference between the predicted hit ratio value and the result of this simulation (the actual result of the simulation is not shown explicitly, but can be obtained by subtracting the error from the predicted value). The figure also shows the maximum difference, in the range $20 \leq T \leq 60$, between the predicted and actual hit ratios for each case. The errors in the predicted hit ratio never exceeded 1.4 percentage points for any of the test cases. Note also that the actual hit ratio is always slightly *better* than the estimated hit ratio. Thus, the prediction obtained from (6) is not only consistently accurate, but is also slightly conservative.

6. CONCLUSION

The main conclusion of the paper is that the hit ratio of each pool in a mixed cache can be estimated as an independent function of the average cache residency time. In addition, the cache residency time is closely related to the cache memory size due to Little's law.

The ability to treat the pools in a mixed cache independently is of great practical interest, and it is also satisfying from a conceptual standpoint. It clarifies the link between hit ratio and locality of reference. Since locality of reference is a characteristic of individual data sets and collections of data sets, the behavior of the hit ratio should be also. We have shown that the hit ratio can, in fact, be analyzed in this way.

The key to the approach is that hit ratio must be treated as a function of the average time which a track spends in the cache. Due to the LRU mechanism, and also due to the tendency of I/O to a given track to come in bursts, the average times-in-cache of multiple pools sharing a cache tend to be close to each other -- close enough so that for practical purposes, the differences between them can be neglected. Thus, the hit ratio of each pool in the cache can be treated as an independent function of the average cache residency time.

This situation leads to a simple and practical scheme for comparing large numbers of possible cache configurations so as to better understand the potential benefits which might result from the use of cached controllers. As shown by the empirical results which we have obtained, the hit ratio estimates which the technique provides are very accurate.

REFERENCES

1. C. P. Grossman, "Cache-DASD Storage Design for Improving System Performance," IBM Systems Journal, v. 24 nos. 3/4, pp. 316-334, 1985.
2. K. G. Dahman, G. L. Grossman, "Effective Use of Cached DASD in a Data Base/Data Communication Environment," CMG Proceedings pp. 425-431, 1983.
3. A. J. Smith, "Disk Cache - Miss Ratio Analysis and Design Considerations," ACM Transactions on Computer Systems, v. 3 no. 3, pp. 761-203, August 1985.
4. A. L. Bastien, "Cached DASD Performance Prediction and Validation," CMG Proceedings, pp. 174-177, December 1982.
5. J. P. Buzen, "BEST/1 Analysis of the IBM 3880-13 Cached Storage Controller," CMG Proceedings, pp. 156-173, December 1982.
6. P. J. Denning, "The Working Set Model for Program Behavior," ACM Communications pp. 323-333, May 1968.

ACKNOWLEDGEMENT

The authors would like to acknowledge the extensive help provided by Mr. J. S. Hyde, the developer of the IBM Cache Analysis Aid, in making possible the results of this paper.

WORKLOAD CHARACTERIZATION FOR SUBSYSTEMS: A CASE STUDY FOR IDMS

Ellen M. Friedman
SRM Associates, Ltd.

ABSTRACT

There has been a significant increase in the number of installations utilizing Cullinets IDMS database system in recent years. Most of these installations use CICS/VS for data communication and interface with IDMS/DB through an SVC. Unfortunately, the available techniques for analyzing IDMS performance and capacity issues has not kept pace with this growth in new applications.

This paper presents a methodology for workload characterization for subsystems with the particular emphasis placed on IDMS applications. We will discuss the data collection and analysis tools available for performance analysis and workload characterization studies. In addition, factors critical to IDMS on-line performance will be highlighted and the differences between using CICS and IDMS for data communication will be reviewed. A case study detailing the methodology and analysis process is presented.

Introduction

The number of installations utilizing IDMS/DB has increased significantly in recent years. However, the information concerning IDMS performance analysis and capacity planning has not quite kept pace with this increased usage.

IDMS, like other TP systems is not necessarily comprised of a set of transactions or run-units which are homogeneous with respect to their consumption of computer resources. Therefore, we propose a methodology for profiling transactions/run-units based on resource utilization parameters to facilitate the construction of homogeneous workloads. The methods and techniques we will discuss in this paper can aid the capacity planner to better understand resource usage for any subsystem and in particular, for IDMS workloads.

Yogi Berra once expressed, "You can observe a lot by watching". In other words, in order to know what's happening in the subsystem you have to look at it. Workload characterization is the process of analyzing a set of data in order to understand the relationship between resource usage, transaction arrival rate and workload mix on system performance. It is the process of forming homogeneous workloads consisting of similar transaction/run-unit resource profiles. The results of such studies will enable the analyst to formulate an accurate and meaningful representation of the workloads comprising the system.

Background- IDMS/R

IDMS/R, a Cullinet relational DBMS, manages the flow of information between application programs and the database. It can operate in both "Local" and "Central Version" mode. While executing in Local mode, only one application program at time can access the database. The Central Version (IDMS-CV) provides multi-programming capability; allowing multiple concurrently executing application programs to access the database using a single copy of the DBMS. As a result, the use of IDMS-CV is preferable in most environments. Central Version operations are also required for application programs which execute under a teleprocessing system (TP). If a TP system other than IDMS-DC is used, an interface module tailored to the specific TP monitor is set-up to be shared by all teleproccessing application programs.

IDMS-DC is an extension of the IDMS-CV providing all of the CV capabilities, as well as controlling the data communications environment which maintains the teleprocessing services necessary for on-line transaction execution. In addition, a Universal Communications Facility (UCF) is provided which allows on-line application programs to be executed from terminals controlled by a TP monitor, while providing all the services of IDMS-DC. UCF is an interface that supports CICS, SHADOW, INTERCOMM, TASK MASTER, WESTI, IDMS-DC, TSO AND CMS.

For the purposes of detailing the workload characterization methods, we will restrict the discussion to the IDMS-CV (the multiprogramming version), and consider both IDMS-DC usage and CICS-DC with the UCF interface [Ref. 1,2,3,4].

General Methodology for Workload Characterization- Subsystems

The workload characterization process requires a representative sample period and relevant performance measurement for analysis. The time varying nature of workload resource usage, and the dependencies upon particular workload mix and arrival rates which impact system performance must be considered as part of the characterization process.

For MVS systems, resource consumption information can be determined on a performance group basis from RMF/SMF data. Average resource profiles can be constructed for each region/performance group in terms of their relative usage of CPU, I/O and memory service. By examining graphs of these average resource profiles over time, we can determine which time periods to use as a sample for a more detailed characterization analysis. In the case study, we discuss the use of such graphics as part of a general data gathering and knowledge acquisition process which is key to developing a realistic and representative characterization of the workloads.

In studies of capacity planning, forecasting and analytic modeling efforts are usually performed for "peak periods". A determination of "typical or representative sampling intervals can be made by examining the sampled distribution of several critical performance variables. Extreme intervals, or intervals containing performance exceptions should be eliminated when performing studies of workload characterization.

When the sampling process is complete, transaction profiles based upon resource consumption can be developed for the selected samples. The construction of transaction profiles is highly dependent upon the particular subsystem we are examining. In a previous paper, we outlined some of the significant variables critical to CICS/VS performance which should be used when classifying CICS work [Ref. 5]. The process of determining those variables which are critical to

performance and are also indicative of resource usage may not be as obvious for all subsystems. Generally, we determine resource usage based upon CPU, and some indication of memory and physical I/O counts. Later in this paper, we will discuss which performance variables we found to best define resource usage in our characterization of IDMS.

Cluster Analysis

The objective of the workload characterization process is to define those key elements which accurately describe the resource consumption of the workload. The general classification approach is to form workloads that are mutually exclusive and internally consistent; that is, the variability within a workload or group should be less than the variability observed between workloads or groups. One popular approach to the problem of forming homogeneous workloads is to utilize cluster analysis based upon a subset of the resource consumption variables.

Given a number of objects, each of which is described by a set of numerical measurements, the objective of cluster analysis is to devise a classification scheme for grouping the objects into a number of classes such that objects within classes are "similar" in some respect and unlike those from other classes. In the case study, we will utilize clustering techniques to form groups of transactions/run-units using their description as defined by the resource consumption variables. The initial choice of the particular set of measurements used to describe each observation constitutes the frame of reference within which to establish the clusters. The choice of measurements reflects the analysts judgment of their relevance to the classification process.

There are several important decisions to be made before inputting a set of data to a clustering algorithm these include:

- How many variables are desirable and practical?
- Which variables to include?
- Are the variables correlated with each other?
- Should the data be standardized in some way to minimize the impact of measures on different scales?
- What is the optimal number of clusters?

It is quite apparent that the choice of input variables will determine the resulting classification, and different groupings may arise as a result of using unstandardized rather than standardized data. The variables are usually standardized, since many of the clustering algorithms are sensitive to variables measured on different scales. A potential danger in using these algorithms, without first adjusting the variables to the same scale, is to artificially give greater weight to those measures (variables) exhibiting greater variability in the algorithm (e.g. storage measured in K Pages vs. CPU measured in msec). In addition, determining the optimal number of clusters for any given data set remains an outstanding problem in the application of clustering techniques. There is a vast amount of literature which discusses these issues. A good introduction can be found in Brian Everitt's text entitled Cluster Analysis [Ref. 6], as well as a brief overview in the SAS Statistics User's Guide [Ref. 7]. In the case study, we will discuss the techniques used to determine homogeneous workloads; including a discussion of sample selection, variables selection, standardization issues and determining the number of groups. We will also review the SAS procedures employed in our analysis. In the case study, we used FASTCLUS, CLUSTER, canonical discriminant analysis and principle components to aid in the determination of the grouping of run-units.

Workload Characterization for IDMS/CV

Data Collection

At the performance group/region level, RMF/SMF data can be used to obtain overall resource consumption information. In addition, the IDMS/R performance monitor (IDMS/R-PM) can be used to capture and record task performance information for IDMS-DC/UCF tasks, an ADS/On-Line dialog, an External Run Unit (ERUS) or CICS transactions external to IDMS-DC-UCF. The IDMS/R performance monitor records can be stored on the SMF file as a user SMF record type NNN (usually written as an SMF 230 record). The SMF header contains the SYSTEM ID, and date-time stamp information. The RT1 record (also referred to as the IDMS/R-PM record) contains information concerning database activity, program type (IDMS/DC , CICS etc.), terminal activity, and CPU and memory resource usage by task/run-unit [Ref. 1]. The IDMS-DB/DC Transaction Statistics can be found in Table I.

IDMS Workload Analysis

To properly characterize the IDMS/CV, we must profile transactions or run-units in terms of their usage of memory, CPU requirements and database activity. In addition, we must first qualify whether the program or dialog was executed under IDMS/DC-UCF, CICS-DC, Batch or some other TP monitor. We must also distinguish between internal IDMS-DC and other TP monitor system tasks, and user tasks. Since ADS/On-line dialogues are inherently more resource intensive than normally generated Cobol or PL/1 programs, we should further identify the run-units according to the DC tasks, first-level programming language.

There is an inordinate number of statistics collected for each IDMS transaction; however, not all of these are immediately concerned with resource consumption nor are they all useful as performance indicators. We will discuss that subset of variables which we have found to be useful in characterizing the transaction set for the case study. In a different IDMS environment, the variables which we excluded from our analysis might have played an important role in characterizing the system. This is due to the fact that the workload characterization process is environment dependent. Generally, however, we characterize the transactions/run-units based upon measures of CPU, I/O and memory usage. A brief discussion of the performance concerns for IDMS systems can be found in the IDMS-CV/DC Systems Operations Guide [Ref. 2, Section 5].

In order to characterize the IDMS system in terms of resource usage, we need to examine CPU consumption, memory or storage utilization and some indication of logical and physical I/O usage.

CPU

Using information from RMF, we can determine the total CPU utilization for a particular performance group/IDMS region. In addition, the IDMS performance record (SMF 230) provides the total TCB time used by each run-unit. This time totaled across all run-units should match the TCB time recorded in the RMF TYPE 72 record for the entire region. Unlike CICS monitor data, the SMF 230 records reports on both task or user TCB time, as well as the amount of system time (including scheduling, terminal handling etc.) used by the run-unit. It should be noted however, that this decomposition of System CPU Time and Task CPU Time is only recorded if a parameter in the SYSGEN is set. Otherwise, all of the TCB time for the task is reported as System CPU time. cIn the case study, the task TCB time is not decomposed and the total TCB time is recorded as System CPU.

I/O

The SMF TYPE 30 records provides the total number of EXCP's used by each performance group/region. The IDMS performance record additionally provides the number of physical I/O's for each run-unit. The total I/O count for each run-unit = the Number of Page Reads + the Number of Page Writes. In addition, the record contains the number of records requested, the number of database calls, the number of pages requested, an indication of poor page size and buffer management, and the number of database locks held by the task.

In order to determine the best subset of these variables to utilize in the classification process, we examined the distribution of each of these variables and examined their relationship with CPU time and elapsed time. Since, we did not want to include variables which were highly correlated with each other, we also reviewed their interdependencies and correlations (specific details are discussed in the case study).

Memory/Storage

From RMF, one can compute a memory occupancy factor (total number of pages used based on MSO service units) for each region/performance group. In addition, the IDMS performance record gives an indication of each run-units usage of storage. The measures include: the amount of storage used from the storage pool (IDMS/DC storage), the amount of storage used from the program pool, and the amount of storage kept from the storage pool. In addition, the number of storage GETS and Puts is recorded. Since storage is also affected by the use of many programs and whether or not those programs are already resident, we also must consider the number of programs called and the number of programs loaded.

Other Classification Variables

Other performance variables may also be used in the classification process to distinguish one set of transactions (run-units) from another. For example, the number of bytes read/written from/to the terminal is an indicator of line utilization. Further, the number of levels doing DB work can be used as a measure of the level of program complexity.

We also examined scatter plots of these variables against elapsed time and CPU time, and their correlation with other indicators of resource usage.

Other IDMS Performance Concerns

The size of each program pool will determine the amount of storage available to the IDMS-CV/DC for loading non-resident modules. The maximum number of tasks and external run-units active and the size of storage pools will impact the total amount of storage required.

The number of database key locks maintained by an IDMS system can impact CPU usage. The efficient use of buffer pools will also contribute to system performance. Factors often considered are the ratio of PAGES REQUESTED/ PAGES READ (logical to physical I/O ratio), the ratio of RECORDS REQUESTED/ NUMBER OF RECORDS CURRENT FOR RUN UNIT; these ratios should not be that high.

In any on-line database system, the database and DASD configuration can tremendously impact the resulting performance. In an IDMS/R database structure, the database is stored physically as one or more data sets on DASD and can be accessed using BDAM, DAM, VSAM or UPAM [Ref. 3]. Each database area corresponds to a portion of a file stored on DASD. These areas defined in a database can be such that either a) all areas map to a single file, b) each area can be mapped into a different file, or c) one area can be mapped into several files. As a result, the number of EXCP's recorded for a particular file/DDNAME may be a result of database usage from a single area or from a portion of many different areas. It is therefore difficult to tune the I/O subsystem based solely on RMF/SMF DASD and file utilization information.

The database structure and the resulting number of physical I/O's will depend upon how the database administrator has mapped these areas to DASD. As such, to minimize any I/O bottlenecks, the database administrator or performance analyst should attempt to correlate the database statistics provided by the IDMS performance monitor with the DASD and file utilization information that can be derived from RMF/SMF.

Case Study

Overview

The formulation of an accurate and meaningful representation of system workloads is a desired goal of the capacity planning process. The following case study serves to exemplify the workload characterization process. The analysis was performed using graphic techniques, by examining the distribution of the important performance variables, and using cluster analysis methods.

Description of System

As a case study, several days of RMF, SMF and IDMS performance measurement data were examined for a large IDMS system environment. The CPU was a 3090-200 with 64 megabytes of real memory and 64 megabytes of expanded storage executing under MVS/XA. The processing environment consisted of 5 large production regions running under Release 10.0 of IDMS/R , one stand-alone production CICS region and some minimal batch work. The daily peak processing hours for the system were between 11:00-12:00 in the morning and 1:30 to 5:00 in the afternoon. During these hours the system was approximately 85% busy, with the majority of the activity attributable to three of the IDMS regions.

Data Analysis

Workload Characterization by Region

The RT1 performance reports were reviewed weekly, for each of the IDMS regions to determine the average number of transactions processed per hour, average response times and average transaction resource usage by region. In addition, we examined the resource usage of individual IDMS transactions to determine whether they constituted a homogeneous group.

In Figures I and II, we have included a decomposition of the CPU busy by each of the major workloads for two different sample periods. Average CPU busy was graphed hourly to display the variation in utilization across the on-line day. The data indicates that the peak processing time is fairly consistent between 11:00-12:00 and 1:00-5:00 P.M.

Using data stored in a SAS database, in conjunction with overall IDMS workload statistics, we were able to identify typical average and peak resource usage patterns. Graphs comparing the average resource profiles of each of the IDMS regions (constructed for peak hours) can be found in Figures III and IV. These profiles are defined in terms of average CPU, and DASD service requirements. The service time profiles are measured in terms of milliseconds per transaction and do not include queuing delays.

An examination of these graphs suggests that the IDMS1 (I1) region is significantly different from the others. The CPU requirements per transaction are almost twice that of the other regions (Figure V). A comparison of the average resource profiles constructed for each of the regions between September and December indicates a change in decomposition. This suggests that changes in workload mix and arrival rate may have been responsible for the differences observed in average resource profiles.

The graphs we have selected are only a sample set, other periods of time were also examined to determine whether an "average transaction" could be constructed to serve as a typical measure for use in trend analysis and analytic modeling. Since the data suggested variability in average resource usage for each of the regions, it was necessary to further decompose each region in terms of groupings of homogeneous run-units/transactions. Using average profiles constructed for these groups, trending and analytic modeling could be performed with greater accuracy.

Workload Characterization- Detailed

A sample peak period consisting of the hours of 1:00-5:00 P.M. on two separate days was selected to analyze the average resource consumption of each of the run-units. For illustrative purposes, we have reduced the problem to forming homogeneous workload groupings for only one of the IDMS regions (IDMS1).

The distribution of all resource parameters critical to performance was examined for the region. In addition, we examined the relationship between these variables and both CPU consumption and elapsed time. The variables we examined included all of the database statistics, and terminal, storage and program information. We also examined the correlations between these variables, so that the total number of variables could be reduced to a set of uncorrelated variables containing the same level of information as the original. The preliminary subset of variables we examined included:

- CPUtime
- Number of Physical I/O's
- Number of Records Requested
- Number of Pages Requested
- Number of Database Requests
- Length of Terminal Read (in bytes)
- Length of Terminal Write (in bytes)
- Number of Programs Loaded
- Number of Programs Called
- Number of VIA records with Oflow
- Number of VIA records without Oflow
- Amount of Storage Used from Storage Pool
- Amount of Storage Used from Program Pool
- Number of Levels doing DB work
- Number of Records Current of Run Unit

Based upon the resource information from this preliminary set of variables, the task is to form a set of homogeneous groupings of transactions/run-units. The first step was to decompose the region by program type, as we were not interested in classifying system transactions, and resource intensive ADS/ON-LINE transactions with the general population of user tasks. Given this initial decomposition, we used a combination of clustering techniques, and principal components to form groups of user tasks/run-units. Once these clusters were formed, we used canonical discriminant analysis to describe the groups in terms of the original variables.

Overview of the SAS PROCEDURES

In the case study, we utilized FASTCLUS, CLUSTER, canonical discriminant analysis and principle components to aid in determining the grouping of run-units. The SAS FASTCLUS procedure finds disjoint clusters of observations using Mac-Queen's K-Means algorithm. It is designed to handle very large data sets with relatively few passes through the data. There are many other programs available to perform cluster analysis, many of which are published [Ref. 10]. We have chosen to restrict our discussion to the SAS procedures, as they are more widely available to the data processing community and the newest release of SAS (1985) has been updated to reflect recent advances in the field.

The FASTCLUS procedure requires the user to input the maximum number of clusters or groups to be classified. Since one may not know how many groups optimally describe the data points, this process could require several iterations. At the end of each iteration, the clusters are evaluated by certain "Goodness-of-Fit" criterion. As output, FASTCLUS provides several statistics for each variable including: an R-Squared value for predicting the variable from the cluster, and the ratio of between cluster variance to within cluster variance. In addition, it provides a statistic called the Cubic Clustering Criterion (CCC) which can be utilized to determine the number of clusters. (Note: This statistic is an approximation to the expected value of R-Squared and SAS indicates that the formula they used to compute it may change in future releases).

At the end of each run, the CCC and R-Squared values are recorded along with the number of clusters. The analyst typically will run the program for cluster groups ranging from 1 through 25. Alternatively, the analyst can run the CLUSTER procedure which automatically produces the CCC and R-Squared statistics for values of one cluster up to as many as desired, in a single run. Plots of both of these statistics can be generated to determine the number of groups. When the Cubic Clustering Criterion is plotted against the number of clusters, peaks on the plot greater than 2 or 3 indicate good clusters. The Cluster procedure also prints pseudo F and T^2 statistics. These values can also be plotted against the number of clusters, peaks on the plot are an indication of the number of clusters yielding good separation. The pseudo T^2 statistic can be used by looking for large values and taking the number of clusters to be one greater than the level at which the peak occurred.

The first stage in many cluster analyses is to convert the raw data matrix into a matrix of inter-individual "similarity", or distance measures (i.e., measures of how alike the groups of transactions are). Using FASTCLUS, the clustering is done on the basis of Euclidean distance measures computed from the variables.

The initialization routine used by FASTCLUS makes it sensitive to outliers. As such, it can be an effective method for detecting outliers, as they often appear as clusters containing very few members. Once the outliers are removed from the data set, the FASTCLUS procedure can be reinvoked with a reduced number of clusters. The goal is to choose the smallest number of groups which describe the data with greatest accuracy.

The test statistic, the CUBIC CLUSTERING CRITERION requires the variables to be uncorrelated. If the variables are correlated, we must first perform a principle component analysis to reduce the dimensionality of the problem. Each principle component is defined as a linear combination of the original variables and they are jointly uncorrelated.

The FASTCLUS procedure was utilized to form a preliminary set of clusters. The Cluster procedure was then used to hierarchically cluster the observations. The clustering methods available are average linkage, the centroid method, complete linkage, density linkage, maximum likelihood for mixtures of multivariate normal distributions with equal variances but possibly unequal mixing proportions, the flexible-beta method, McQuitty's similarity analysis, the median method, single linkage, two-stage density linkage, and Ward's minimum variance method. All of these methods are based on agglomerative hierarchical clustering procedures [Ref. 10].

Using agglomerative hierarchical clustering methods, each observation initially begins in a cluster by itself. The two closest clusters are merged to form a new cluster which replaces the old cluster. The process of merging clusters is repeated until only one cluster remains. The major difference between the clustering methods is in the choice of the distance measure.

In the case study, we used average linkage, density linkage and Ward's minimum variance method. In average linkage, the distance between two clusters is defined as the average distance between pairs of observations, one in each cluster. The density linkage method (Wong and Lane 1983), refers to class of clustering methods using nonparametric density estimates. The distance measure, or dissimilarity measure is defined as the reciprocal of an estimate of the density midway between two adjacent observations x_i and x_j. We used Wong's hybrid clustering method which uses density estimates based on the preliminary cluster analysis from FASTCLUS (a k-means method). Using this method, two clusters C_k and C_l are considered adjacent, if the midpoint between their cluster means $\bar{x}_k$ and $\bar{x}_l$ is closer to either $\bar{x}_k$ or $\bar{x}_l$ then to any other cluster mean. As output, the density linkage method will print the number of modal clusters, where the number of modal clusters formed is an indication of the optimal number of clusters or groups.

Cluster Analysis Process for the IDMS workload

To avoid the potential problem of using variables measured on different metrics, we standardized each of the variables to a mean of 0 and standard deviation of 1. (This standardization is often referred to as forming a Z-Statistic and can be easily accomplished using PROC STANDARD). The standardized data was used in all subsequent analyses.

Using the SAS FASTCLUS procedure, we formed an initial grouping of run-units. Our objective in this first run was to remove the presence of outlier observations, where we defined outliers as clusters containing less than 5 observations. Given a reduced set of observations, we could then proceed to determine the optimal number of classes to best describe the set of run-units.

A preliminary set of clusters was produced using FASTCLUS with a maximum number of clusters equal to 25. The cluster summary statistics from this initial run can can be found in Table II and Table III. The information displayed in Table II was used to identify those groups with fewer than 5 observations. The data presented in Table III provides summary statistics for each of the variables including the R^2 value for predicting the variable from the cluster, and the ratio of between cluster variance to within cluster variance. After removing the outlier values, we reinvoked the FASTCLUS procedure with 22 clusters and observed that one of the groups, "Cluster 20" was significantly different from the others (TABLE IV). Therefore, since we had already confirmed that this group was significantly different from the other's, we deleted it and reinvoked the FASTCLUS procedure with 21 clusters in order to pursue the identification of differences amongst the remaining observations.

An examination of the correlation matrix suggested that many of the variables were correlated. We therefore had several options in our analysis. One option was to examine the principle components and select those variables with high loadings (i.e. those variables corresponding to large coefficients for the eigenvectors) that are also uncorrelated with the other variables selected. As a second option, we could choose the first few principle components which explained over 80% of the variance as input to the cluster procedure. (The principle components are linear combinations of the original variables and are jointly uncorrelated).

To summarize up to this point, there are several factors to consider in the analysis, one is to remove the presence of outlier observations which can obscure any underlying differences between observations and groups. The second critical factor is to identify the important variables to include in the cluster analysis. In our analysis we actually went through several iterations, "boot-strapping" our way through the problem of identifying groups and variables. We used an initial set of variables as input to the FASTCLUS procedure and identified the outlier values. Once the outliers were removed, we were able to concentrate our analysis on identifying the important variables. The selection of variables is a critical step in forming homogeneous groups of observations. Not all of the steps in our analysis have been detailed, we have included the major portions to highlight the process.

In Table V, we have included the principle components and their coefficients. The first principle component has high positive loadings for data base usage, the number of physical I/O's and CPU. The second principle component exhibits high positive loadings for storage usage, both program and storage pool, and on the number of programs called. The first five principle components account for 80% of the variation.

As a result of the various analyses we performed, the following variables were selected as input to the CLUSTER procedure:

- CPUtime
- Number of Physical I/O's (IOCOUNT)
- Number of Database Requests (RT1DBDBC)
- Length of Terminal Read (in bytes) (RT1DCTRL)
- Length of Terminal Write (in bytes) (RT1DCTWL)
- Number of Programs Loaded (RT1DCNPL)
- Amount of Storage Used from Storage Pool (RT1DCSUS)
- Number of Levels doing DB work (RT1DCCUR)
- Number of Records Current of Run Unit (RT1DBRCR)

It should be noted that at one point, we also included the number of VIA records with and without overflow. We found however, that the cluster procedures yielded better separation between groups when these variables were left out of the analysis.

We used three different methods from the CLUSTER procedure to hierarchically cluster the preliminary clusters produced by FASTCLUS; Ward's minimum variance method, average linkage, and density linkage. For each of these methods we also employed canonical discriminant analysis to describe the observations in terms of linear combinations of the original variables. Our intent was to be able to visually observe whether the clustering procedures yielded a true separation between groups. We produced a graphic representation of the groups by plotting the first two canonical variables which were linear combinations of the original variables, using cluster number as the symbol (see Figure VI, Figure VII and Figure VIII). At first glance, the plots are

indistinguishable, each looks identical except for the symbol designation of cluster number (note: Cluster 1 is replaced with the letter A and Cluster 10 is replaced with the letter J). The graphs provide evidence for the existence of three non-overlapping groups.

We have included plots of the Cubic Clustering Criterion, Pseudo F and T^2 statistics vs. the number of clusters for Ward's method. The plot of the Cubic Clustering Criterion (CCC) against against the number of clusters suggests a peak at a value of 10 distinct clusters. The plots of the pseudo F statistic and T^2 statistic against the number of clusters indicates a peak value at 2 or 3 clusters. These graphs are displayed in Figure IX and Figure X.

We have also included a cluster summary report for the density linkage method in Table VI. The results of the density linkage method indicated the existence of three modal clusters. A tree diagram displaying the results of hierarchical clustering as a tree structure is included for this method in Figure XI. The tree diagram also suggests a division into three different groups.

Summary statistics for each cluster defined by the density linkage method depicting differences in average CPU, IOCOUNTs, number of data base calls (RT1DBDBC), the amount of storage used from the storage pool (RT1DCSUP), the number of records current for the run-unit (RT1DCCUR), the number of program calls (RT1DCNPC), the number of program loads (RT1DCNPL) and average response time (elapsed time) are included in Table VII. The plot of the first two canonical variables indicates that clusters 21 and 9 are different from the other groups. The cluster means presented in in Table VII indiate that cluster 9's usage of resources is below the average and that most of the performance measures for cluster 21 are also below the average (with the exception of the number of data base calls and CPUtime which are slightly above the average).

During the presentation we will review the clusters and discuss their interpretation in terms of business groupings, types of work processed and resource usage.

Case Study Summary

The results of the various cluster procedures indicate that at least 3 clusters should be formed from the bulk of the data. The Cubic Clustering Coefficient produced for WARD's method peaked at nine to ten clusters. However, even using WARD's method, the plots of the first two canonical variables suggested that only three of the groups were non-overlapping and of any real significance. It is not unusual for the cluster procedures to produce different results nor is it unusual for them to produce groups which are truly disjoint. In many cases the clusters produced will not be separated by a great enough distance and it may be necessary to merge nearby clusters by hand to obtain a satisfactory partition. The question then arises, How do you know when you have a good set of clusters? We have suggested the usage of some graphic methods to aid in determining the optimal number, including the plot of the CCC against the number of groups and graphs of the canonical variables. The density linkage clustering methods will provide an indication of the number of modal clusters produced. During the presentation we will discuss this problem in greater detail.

It is important to consider cluster analysis as part of a set of techniques for use in exploratory data analysis. The value of using cluster analysis is that it may suggest relationships between new combinations of variables and observations that had been previously unnoticed. It must be an iterative process. One has to decide on the choice and number of variables to include as input to the clustering algorithm and one has to examine the validity of the clusters in terms of some outside set of criteria and their interpretive value.

The ultimate criterion for evaluating the clusters is whether these groupings facilitate the achievement of the analysts study objectives. The groups formed after careful systematic analysis should be meaningful to the investigator. For example, can the clusters be defined in terms of groups of transactions which perform similar functions (Update transactions vs. Inquiry), or in terms of groups of transactions which comprise a business function (accounts payable, receivable etc.). During the presentation, I will review some of the criteria and also discuss the "meaning" of the groups formed.

Conclusion

The workload characterization process provides a definition of the business workloads, their usage of data processing resources and service requirements. The process can help us to identify long-running, resource intensive transactions, to determine whether guaranteeing an average on-line host response time less than 1 second is feasible and it helps us to discover that not all of the transactions in the system are similar in terms of the amount of resources they consume. By profiling transactions based on resource utilization parameters and forming homogeneous workloads, we can produce more meaningful performance reports and monitor resource usage more effectively. In addition, using homogeneous workloads as input to both analytic and statistical forecasting models will ensure more accurate predictions of future system behavior.

Our objective in this paper was to describe a general methodology for workload characterization for subsystems, to discuss some of the details involved in performing the analyses, and to apply these techniques to characterize and develop an understanding of resource usage for IDMS systems.

While gaining an understanding of the system workloads, we should also gain better insights about what drives them and how they may change in the future. We can't expect to be able to determine how things will change in the future if we don't understand the present.

References

[1] IDMS/R Performance Monitor User's Guide, TDDB-0225-10000.

[2] IDMS-CV/DC Systems Operations TDVC-0225-100G0.

[3] IDMS-DC Concepts and Facilities TDDC-0100-1001.

[4] IDMS/R Database Operations TDDB-0220-10000.

[5] E.M. Friedman, "Workload Characterization: A Case Study for CICS/VS", CMG XIV, December 1983.

[6] Everitt, B.S. (1980), Cluster Analysis Second Edition, London: Heineman Educational Books.

[7] SAS User's Guide: Statistics, 1985 Edition. SAS Institute.

[8] Friedman, H.P. and Rubin, J. (1967), "On some invariant criteria for grouping data". J. Am. Statistical Association, 62,1159-1178

[9] Gnanadesikan, R. (1977) Methods for Statistical Data Analysis of Multivariate Observations. Wiley and Sons, New York.

[10] Anderberg, M.R. (1973) Cluster Analysis For Applications. Academic Press, New York.

Table I. RT1 Transaction Statistics

DC Statistics

Program Activity

- Number of Programs Called/Loaded (RT1DCNPC, RT1DCNPL)

Terminal Activity

- Number of Terminal Reads/Writes/Errors
- Terminal Read/Write Length (RT1DCTRL, RT1DCTWL)

Memory Occupancy

- Storage Used from Storage Pool (RT1DCSUS)
- Storage Used from Program Pool (RT1DCSUP)
- Storage Kept from Storage Pool (RT1DCSKS)

Scratch Pad Facility Usage

- Number of Scratch Gets
- Number of Scratch Puts
- Number of Scratch Deletes

CPU time

-Task CPU time
-System CPU time
Elapsed Time/ Wait time

Other Application Information

- Number of Levels in Application Structure (RT1DCVDB)
- Number of Levels doing DB work (RT1DCCUR)
- Number of Record Buffer Blocks

DB Statistics

Number of Database calls (RT1DBDBC)
Number of Pages Read/Written (RT1DBPRD)
Number of Pages Requested (RT1DBPRQ)
Number of CALC Records with no Oflow (RT1DBCNO)
Number of CALC Records with OFlow (RT1DBCWO)
Number of VIA Records with no Oflow (RT1DBVNO)
Number of VIA Records with Oflow (RT1DBVWO)
Number of Records Requested (RT1DBNRR)
Number of Records current of Run-Unit (RT1DBRCR)
Number of fragments stored
Number of records relocated

TABLE II
FASTCLUS PROCEDURE

REPLACE=FULL RADIUS=0 MAXCLUSTERS=25 MAXITER=1
CLUSTER SUMMARY

CLUSTER NUMBER	FREQUENCY	RMS STD DEVIATION	MAXIMUM DISTANCE FROM SEED TO OBSERVATION	NEAREST CLUSTER	CENTROID DISTANCE
1	2	8.2E-08	1.5E-15	6	8.5988
2	86	0.2017	2.5141	7	4.5273
3	6	0.6782	2.7081	16	6.7635
4	1040	0.6186	5.9179	12	3.7045
5	44	0.4626	2.9798	7	5.2732
6	14	0.3414	2.6914	1	8.5988
7	36	0.5141	3.2165	2	4.5273
8	58	0.4168	3.4238	25	4.0523
9	13	0.3784	1.5709	3	9.1087
10	962	0.6056	5.3829	19	2.4298
11	1	.	0	1	34.6504
12	1176	0.5136	6.2869	18	2.4693
13	2	0.7149	2.0220	15	19.4139
14	2	6.4E-08	1.6E-15	17	27.2465
15	180	0.1354	5.0948	25	6.5022
16	2	6.0E-09	1.0E-16	3	6.7635
17	1	.	0	14	27.2465
18	2950	0.4802	4.9690	19	2.4124
19	14161	0.5717	4.8766	18	2.4124
20	3	6.0E-08	1.1E-15	15	13.5762
21	1	.	0	11	90.9953
22	59	0.5491	3.3156	23	5.1734
23	15	0.6286	2.5971	22	5.1734
24	12	2.3E-07	7.1E-15	14	30.8651
25	1630	0.5734	5.0775	19	3.3649

TABLE III
FASTCLUS PROCEDURE

STATISTICS FOR VARIABLES

VARIABLE	TOTAL STD	WITHIN STD	R-SQUARED	RSQ/(1-RSQ)
RT1DCSUS	1.000000	0.835289	0.303038	0.434799
RT1DCTRL	1.000000	0.511320	0.738831	2.828939
RT1DCTWL	1.000000	0.641404	0.589040	1.433329
RT1DBDBC	1.000000	0.291363	0.915198	10.792230
RT1DBNTK	0.000000	0.000000	.	.
RT1DCSUP	1.000000	0.786478	0.382114	0.618421
RT1DBPRQ	1.000000	0.315593	0.900508	9.051018
RT1DBNRR	1.000000	0.306460	0.906183	9.659 10
RT1DBVNO	1.000000	0.887757	0.212730	0.270213
RT1DBVWO	1.000000	0.237485	0.943661	16.749762
RT1DCCUR	1.000000	0.743866	0.447255	0.809153
RT1DBRCR	1.000000	0.284671	0.919049	11.353170
RT1DCNPC	1.000000	0.713467	0.491509	0.966605
RT1DCNPL	1.000000	0.516839	0.733163	2.747612
IOCOUNT	1.000000	0.545243	0.703028	2.367315
CPUTIME	1.000000	0.387546	0.849968	5.665263
OVER-ALL	0.968246	0.557339	0.669018	2.021316

PSEUDO F STATISTIC = 1889.17
APPROXIMATE EXPECTED OVER-ALL R-SQUARED = 0.35033
CUBIC CLUSTERING CRITERION = 971.094

TABLE IV
FASTCLUS PROCEDURE

REPLACE=FULL RADIUS=0 MAXCLUSTERS=22 MAXITER=1
CLUSTER SUMMARY

	CLUSTER NUMBER	FREQUENCY	RMS STD DEVIATION	MAXIMUM DISTANCE FROM SEED TO OBSERVATION	NEAREST CLUSTER	CENTROID DISTANCE
	1	491	0.4368	4.9223	2	2.3427
	2	974	0.5323	4.3380	6	1.9429
	3	86	0.1621	1.9787	11	4.3671
	4	15	0.2934	1.2120	19	4.5190
	5	44	0.4567	3.0484	11	5.3533
	6	1785	0.3049	4.3447	2	1.9429
	7	180	0.1339	5.0382	22	6.5748
	8	6	0.4973	3.9656	9	5.9135
	9	3020	0.4004	3.5951	13	2.0573
	10	1081	0.4604	3.6409	9	2.1139
	11	36	0.4938	3.1334	3	4.3671
	12	13	0.1747	0.7250	18	7.9515
	13	9708	0.4666	4.1269	9	2.0573
	14	58	0.4110	3.6574	22	3.9548
	15	14	0.3252	2.6664	18	12.0632
	16	1665	0.5547	3.6540	9	3.7411
	17	223	0.6226	5.0236	1	2.4601
	18	207	0.3137	4.4674	21	4.6526
	19	59	0.5078	3.5382	4	4.5190
***	20	12	1.9E-07	5.0E-15	15	67.3018
	21	1397	0.4554	5.5934	6	3.4947
	22	1368	0.4690	3.3989	10	2.7899

*** NOTE: CLUSTER 20 IS SIGNIFICANTLY DIFFERENT FROM THE OTHERS

TABLE V
PRINCIPAL COMPONENT ANALYSIS

	EIGENVALUE	DIFFERENCE	PROPORTION	CUMULATIVE
PRIN1	5.35456	2.61856	0.356970	0.356970
PRIN2	2.73600	1.11371	0.182400	0.539370
PRIN3	1.62229	0.45206	0.108152	0.647523
PRIN4	1.17023	0.04824	0.078015	0.725538
PRIN5	1.12199		0.074799	0.800337

EIGENVECTORS

	PRIN1	PRIN2	PRIN3	PRIN4	PRIN5
RT1DCSUS	0.170322	0.491482	-.021570	-.118045	-.201932
RT1DCTRL	0.015307	-.088955	0.581333	0.164800	-.010228
RT1DCTWL	-.021429	0.299201	-.194592	0.257917	0.408012
RT1DBDBC	0.389275	-.086635	0.095156	0.121537	0.158984
RT1DBNTK	0.000000	0.000000	-.000000	-.000000	-.000000
RT1DCSUP	0.161467	0.465974	0.056251	-.261931	-.264778
RT1DBPRQ	0.372370	-.210336	-.133905	-.123962	-.061241
RT1DBNRR	0.372637	-.204270	-.115904	-.095431	-.088018
RT1DBVNO	0.189553	-.108515	-.438894	0.492344	0.096862
RT1DBVWO	0.173360	-.261863	-.179189	-.512395	0.222567
RT1DCCUR	0.184259	0.236922	-.362247	0.200628	-.450199
RT1DBRCR	0.355379	-.148298	0.146731	0.130160	-.000095
RT1DCNPC	0.105783	0.310432	0.201966	0.266870	0.384463
RT1DCNPL	0.158806	-.177320	0.327099	0.313742	-.449939
IOCOUNT	0.304105	0.237561	0.226227	-.229441	0.169228
CPUTIME	0.400067	0.087685	0.060706	0.009530	0.212773

Table VI

DENSITY LINKAGE CLUSTER ANALYSIS

HYBRID METHOD

ROOT-MEAN-SQUARE TOTAL-SAMPLE STANDARD DEVIATION = 0.673326

NUMBER OF CLUSTERS	CLUSTERS JOINED		FREQUENCY OF NEW CLUSTER	FUSION DENSITY	MAXIMUM DENSITY IN EACH CLUSTER LESSER	GREATER	
20	OB17	OB16	15381	35.0759	20.8854	190.2	
19	CL20	OB13	16647	21.1527	105.4	190.2	
18	CL19	OB1	17887	18.4432	21.7964	190.2	
17	OB11	OB7	1348	14.5246	5.0416	1934.9	
16	CL18	OB4	18594	13.9183	12.4952	190.2	
15	CL16	OB8	20142	5.4498	3.3319	190.2	
14	CL15	CL17	21490	3.6821	190.2	1934.9	*
13	CL14	OB12	21726	3.1789	12.0483	1934.9	
12	CL13	OB2	21878	1.9496	4.4932	1934.9	
11	CL12	OB9	21922	0.6679	0.6160	1934.9	
10	CL11	OB3	22125	0.5274	963.7	1934.9	
9	CL10	OB14	22142	0.3092	1.9915	1934.9	
8	CL9	OB15	22150	0.2198	0.1217	1934.9	
7	CL8	OB10	22329	0.1490	270.5	1934.9	
6	OB21	OB20	59	0.0668	6.6940	8.1E+08	
5	CL7	OB19	22333	0.0571	1934.9	7.3E+66	
4	CL6	OB5	74	0.0213	32108.6	8.1E+08	
3	CL5	OB6	22347	0.0117	9.4361	7.3E+66	
2	CL3	OB18	22360	.0070795	997484	7.3E+66	
1	CL2	CL4	22434	7.4E-04	8.1E+08	7.3E+66	*

3 MODAL CLUSTERS WERE FORMED

TABLE VII
SUMMARY STATISTICS BY CLUSTER
CLUSTER MEANS

DENSITY LINKAGE CLUSTER ANALYSIS

HYBRID METHOD

CLUSTER	ELAPSED	RT1DBDBC	RT1DCSUP	CPUTIME	IOCOUNT	RT1DCCUR	RT1DCNPC	RT1DCNPL
1	0.7114	0.64079	-0.0825	0.68452	0.58698	-0.05583	0.1048	1.03715
2	0.1318	0.26343	-0.2856	0.16116	0.44167	1.44252	0.0392	0.82232
3	-0.0371	0.42091	-0.1936	0.32247	0.89002	3.57097	-1.0454	0.74004
4	-0.4044	-0.17735	-0.8262	-0.27599	-0.47290	-0.19375	-0.4513	-0.37669
5	0.6544	0.23715	1.9439	0.21157	0.12916	2.05681	-0.9192	0.92777
6	0.1517	0.09732	-0.3233	-0.16565	0.05178	-0.72600	-0.4173	3.49086
7	0.6646	3.35073	-0.3732	3.60221	8.14525	-0.78024	-0.8020	-0.22642
8	0.2507	-0.24393	-0.2432	-0.31449	-0.23673	-0.74845	-0.1507	0.02067
9	-0.0894	-0.08418	-0.5746	-0.19650	-0.04959	-0.78024	0.1823	0.88003
10	1.9886	6.64463	-3.2357	1.94430	-0.02396	-0.78024	-2.7131	-0.41421
11	-0.4297	-0.29722	-0.5775	-0.44673	-0.51485	-0.78024	0.0104	-0.32630
12	0.1850	0.17088	-0.4327	-0.04596	0.01855	-0.32191	0.9931	3.29582
13	0.7223	1.58530	-1.0952	1.43638	0.54642	-0.78024	8.7466	3.64877
14	11.0209	0.14175	-0.0117	0.58794	0.52242	-0.78024	-1.0735	2.11769
15	1.0138	1.94082	0.0841	2.06391	1.57714	-0.78024	0.1112	1.04966
16	0.5070	2.41821	-0.0148	2.72968	1.47995	-0.78024	9.0139	1.82641
17	-0.0181	-0.10648	-0.4497	-0.29549	-0.17657	-0.71110	-0.3362	1.73889
18	6.2641	1.20521	0.9758	2.42657	2.30182	-0.78024	4.3520	-0.28807
19	0.0841	0.00785	0.7522	0.20575	0.39443	0.09945	0.6684	-0.50140
20	-0.0889	1.72756	-1.1058	1.57744	0.31745	-0.78024	10.3458	0.05715
21	1.1370	1.23143	-3.2247	0.95404	-0.12677	-0.78024	-2.8109	-0.23153

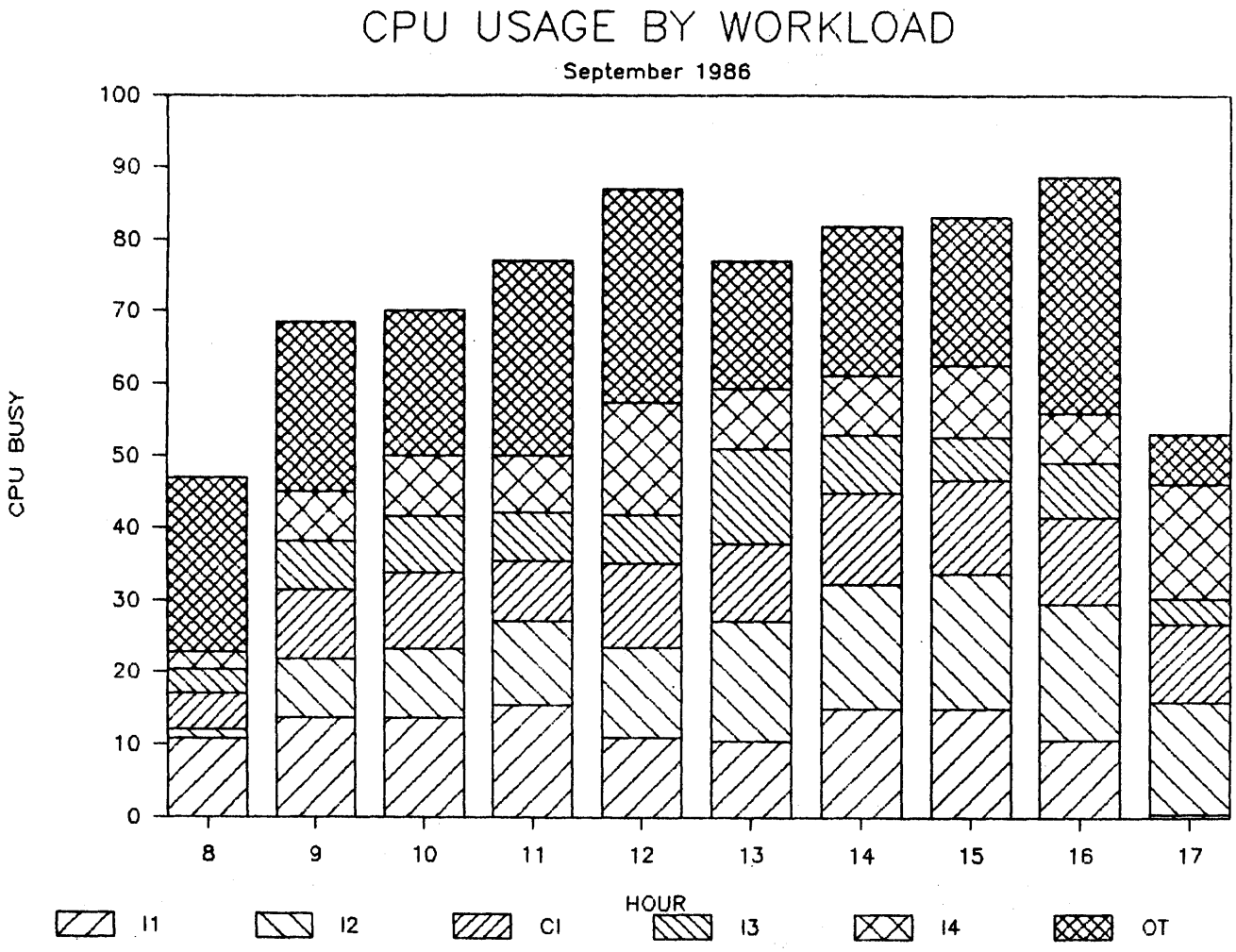

CPU USAGE BY WORKLOAD
September 1986
CPU BUSY
100
90
80
70
60
50
40
30
20
10
0
8
9
10
11
12
13
14
15
16
17
HOUR
I1
I2
CI
I3
I4
OT

Figure I

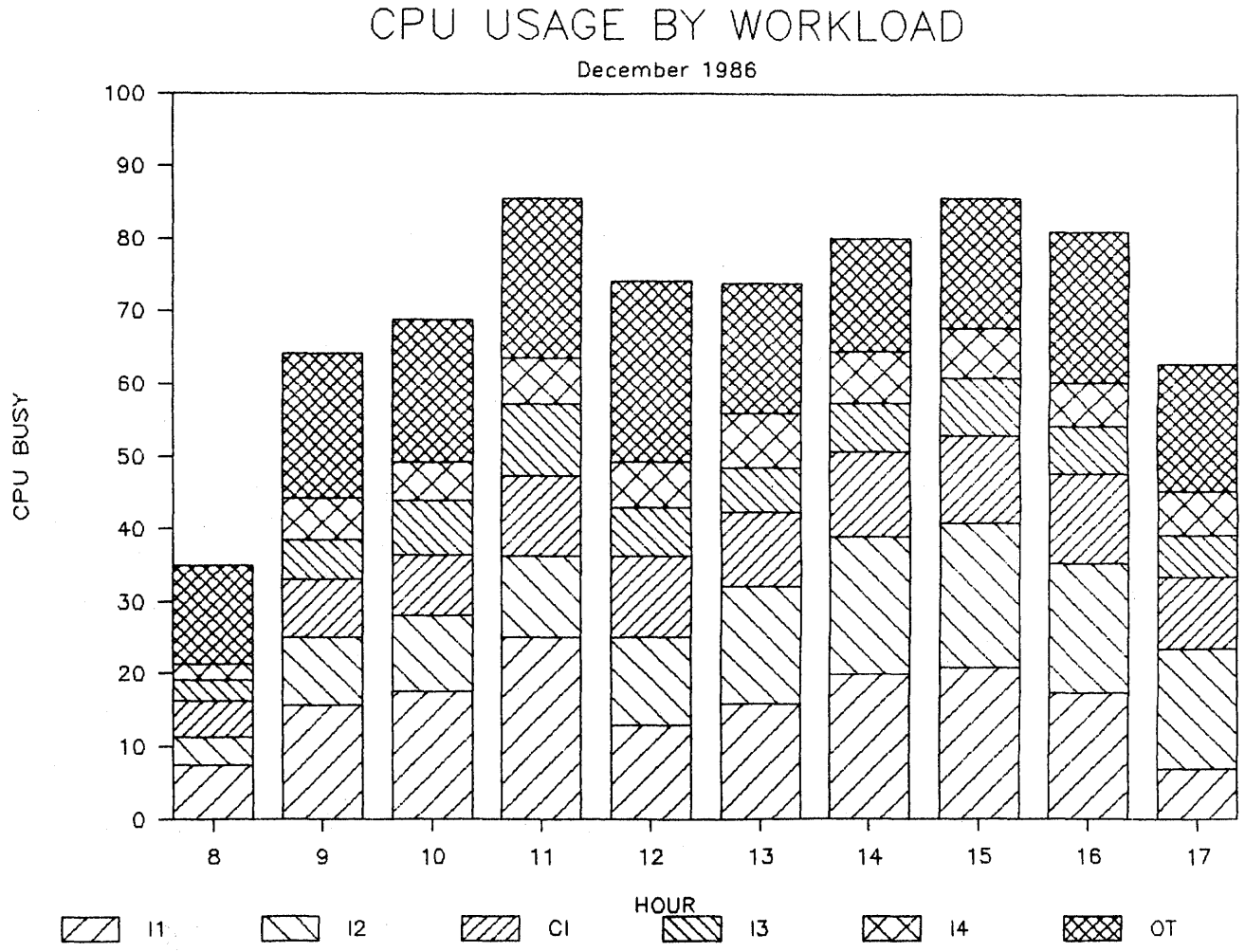

CPU USAGE BY WORKLOAD
December 1986
CPU BUSY
100
90
80
70
60
50
40
30
20
10
0
8
9
10
11
12
13
14
15
16
17
HOUR
I1
I2
CI
I3
I4
OT

Figure II

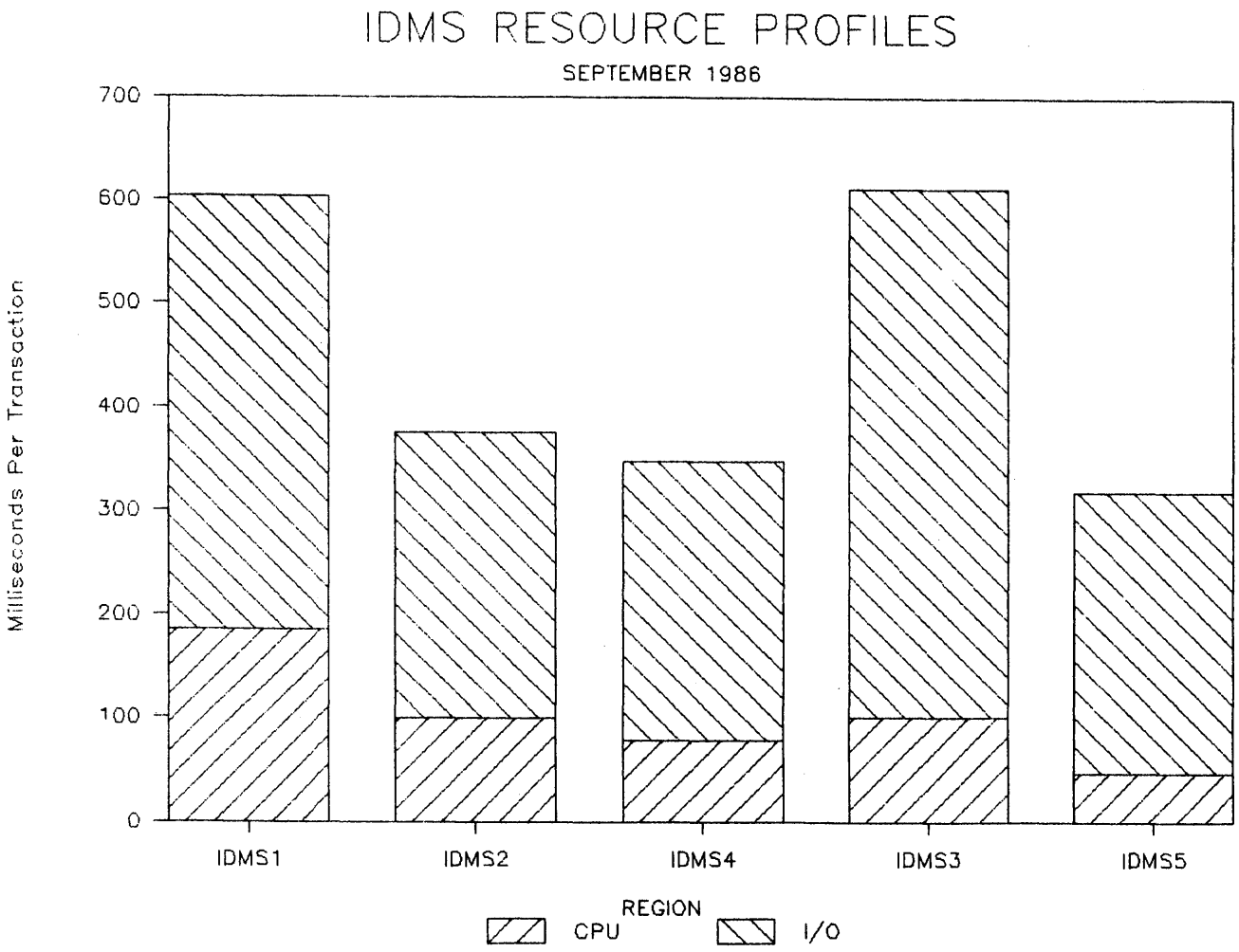

IDMS RESOURCE PROFILES
SEPTEMBER 1986
Milliseconds Per Transaction
700
600
500
400
300
200
100
0
IDMS1
IDMS2
IDMS4
IDMS3
IDMS5
REGION
CPU
I/O

Figure III

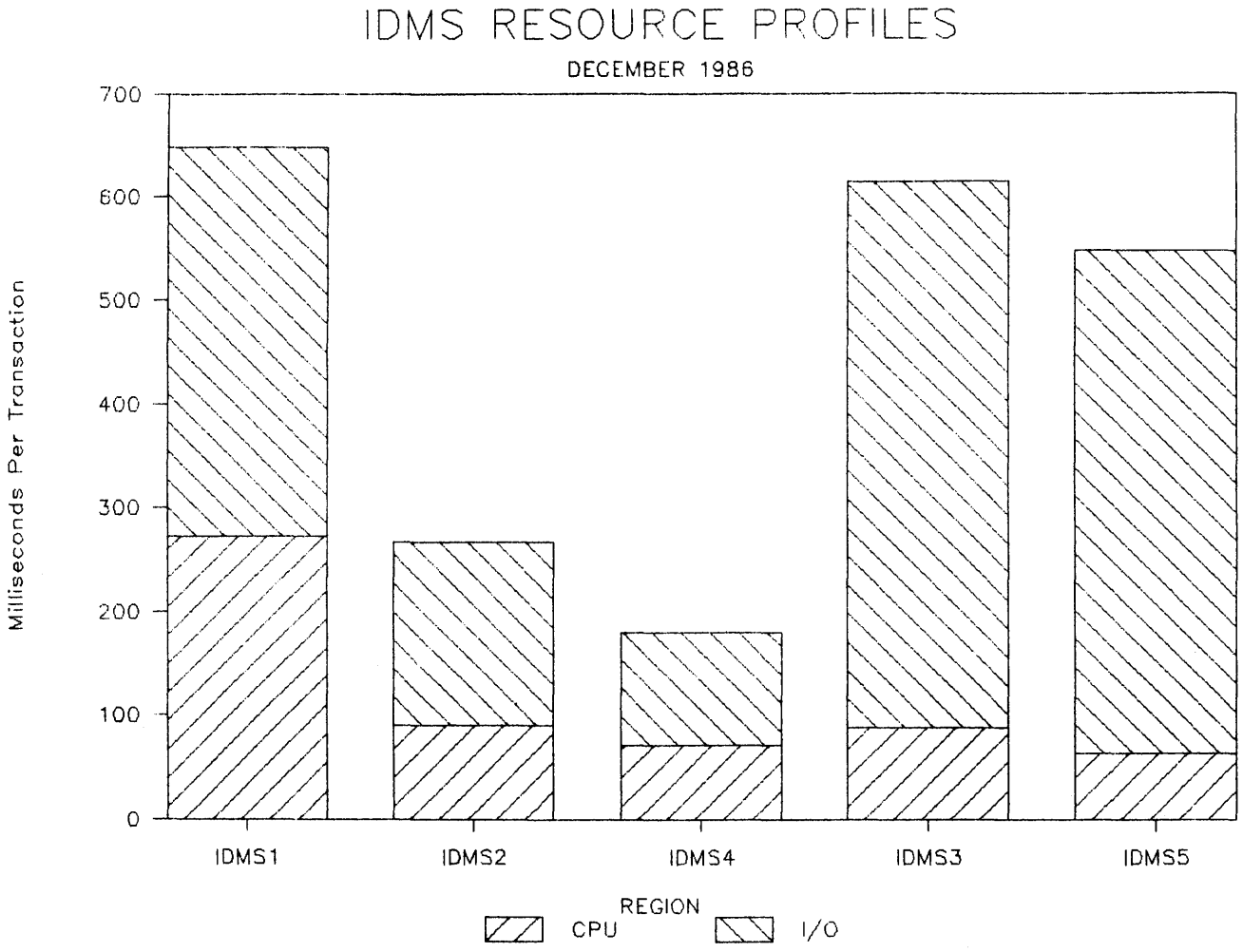

IDMS RESOURCE PROFILES
DECEMBER 1986
Milliseconds Per Transaction
700
600
500
400
300
200
100
0
IDMS1
IDMS2
IDMS4
IDMS3
IDMS5
REGION
CPU
I/O

Figure IV

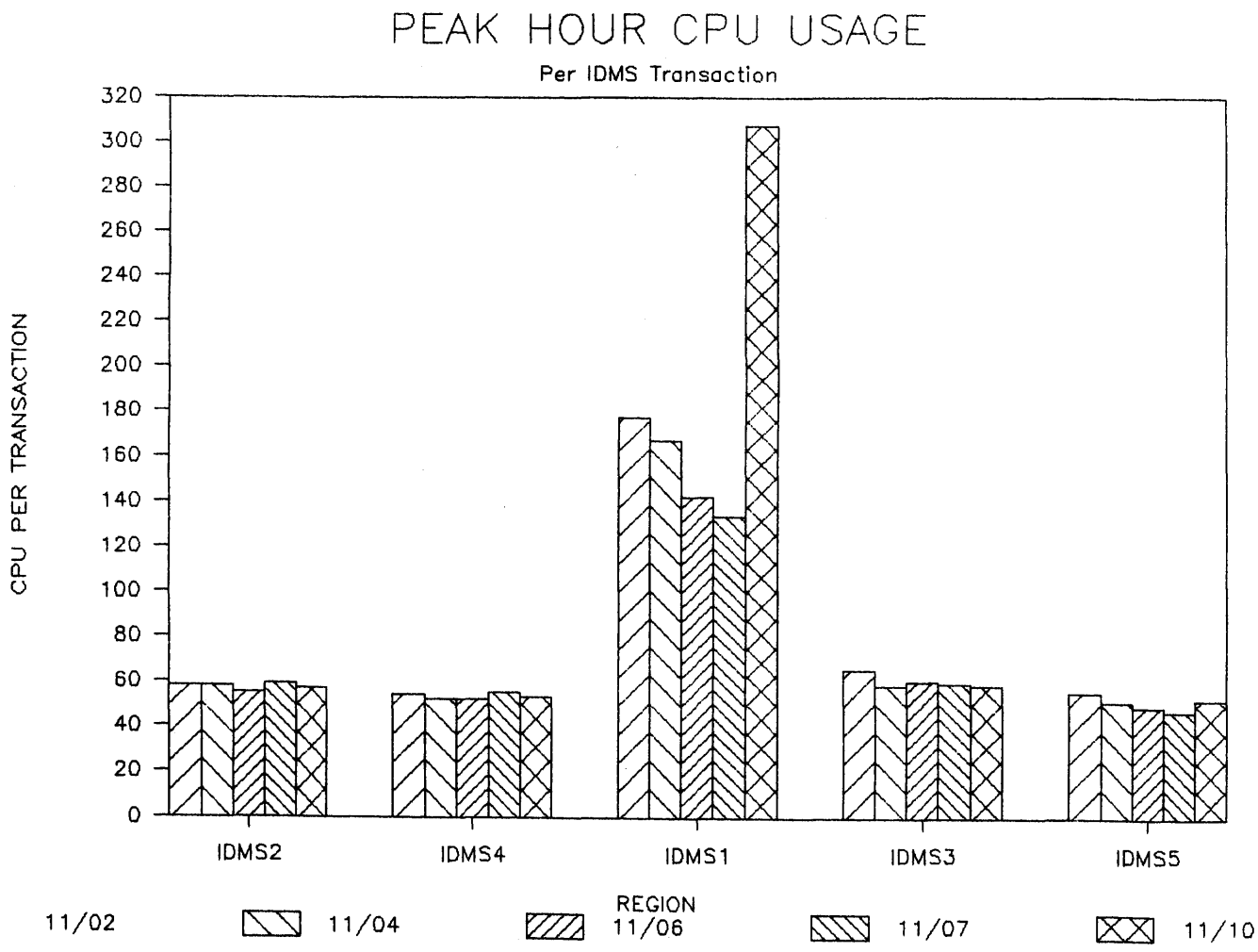

PEAK HOUR CPU USAGE
Per IDMS Transaction
CPU PER TRANSACTION
320
300
280
260
240
220
200
180
160
140
120
100
80
60
40
20
0
IDMS2
IDMS4
IDMS1
IDMS3
IDMS5
REGION
11/02
11/04
11/06
11/07
11/10

Figure V

AVERAGE LINKAGE CLUSTER ANALYSIS
PLOT OF THE FIRST TWO CANONICAL VARIABLES

PLOT OF CAN2*CAN1 SYMBOL IS VALUE OF CLUSTER

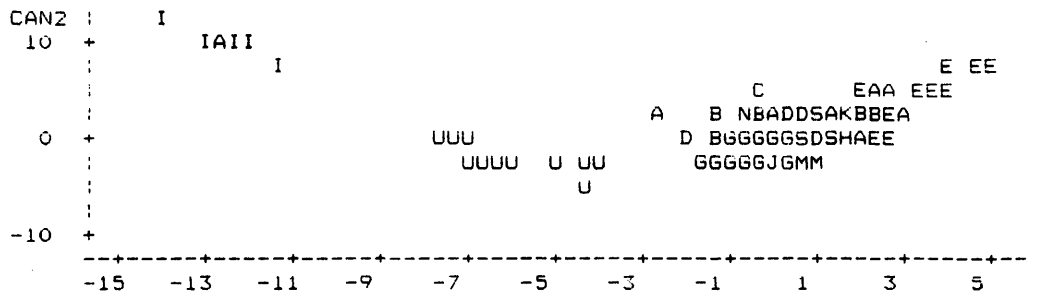
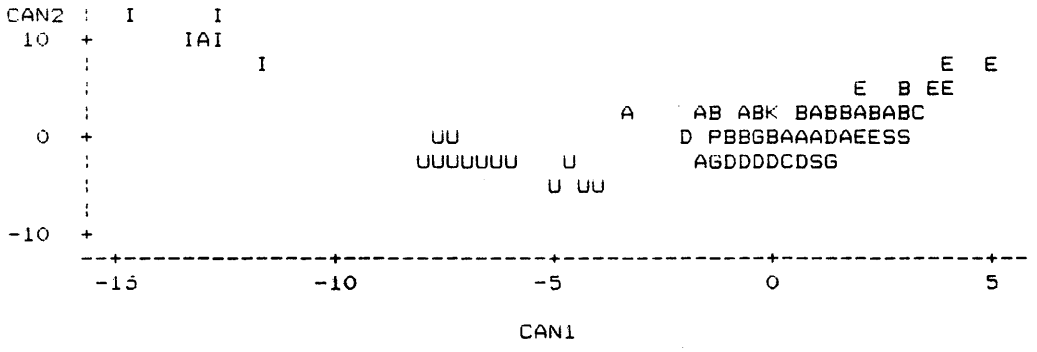


Figure VI

WARD'S MINIMUM VARIANCE CLUSTER ANALYSIS
PLOT OF THE FIRST TWO CANONICAL VARIABLES

PLOT OF CAN2*CAN1 SYMBOL IS VALUE OF CLUSTER



NOTE: 391 OBS HIDDEN

Figure VII

DENSITY LINKAGE CLUSTER ANALYSIS
PLOT OF CAN2*CAN1 SYMBOL IS VALUE OF CLUSTER
PLOT OF THE FIRST TWO CANONICAL VARIABLES

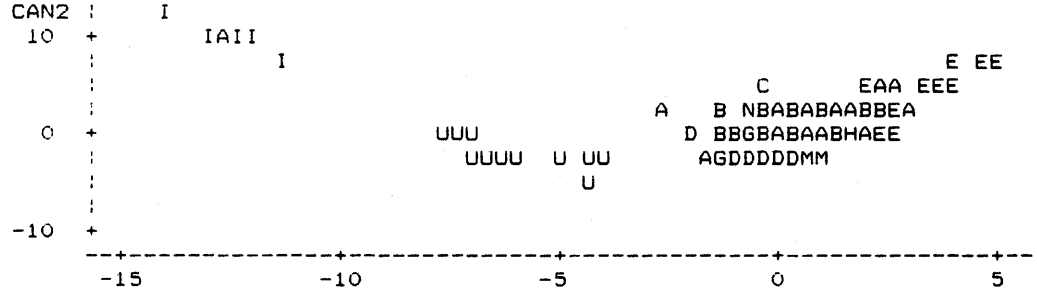


Figure VIII

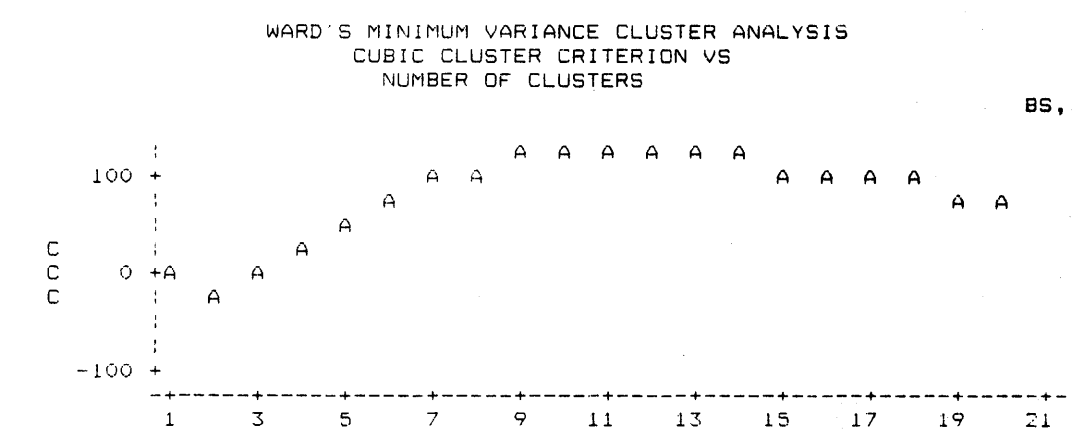


Figure IX

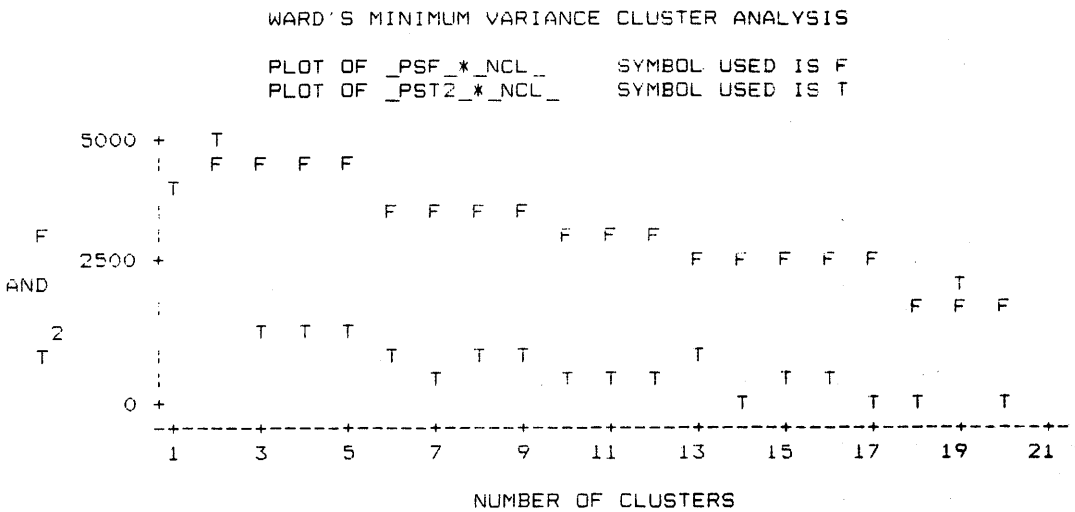

WARD'S MINIMUM VARIANCE CLUSTER ANALYSIS
PLOT OF _PSF_*_NCL_ SYMBOL USED IS F
PLOT OF _PST2_*_NCL_ SYMBOL USED IS T
5000
2500
0
1 3 5 7 9 11 13 15 17 19 21
NUMBER OF CLUSTERS
PSF= PSEUDO F STATISTIC
PST2= PSEUDO T STATISTIC
NOTE: 43 OBS HAD MISSING VALUES

Figure X

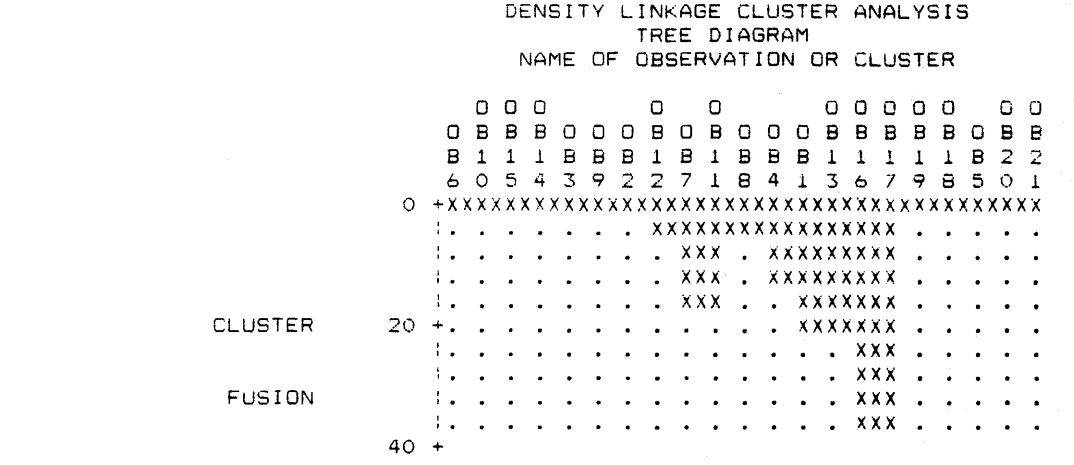


Figure XI

IDENTIFICATION OF TUNING OPPORTUNITIES THROUGH THE USE OF USER BEHAVIOR GRAPHS

Charles E. Hackett
The Guardian Life Insurance Company of America

ABSTRACT

The objective of this paper is to describe a terminal oriented perspective of analyzing on-line system performance. The information derived from this analysis can point the performance engineer toward previously unknown bottlenecks. Case studies are used to examine the analysis technique. The data used is drawn from Landmark's CICS Monitor detail file, and is transformed in custom BAL programs. SAS procedures are used to report the information, which is then used to chart User Behavior Graphs.

I. INTRODUCTION

The need to identify critical CICS transactions can be satisfied by simple criterial. This includes selection by: executed most often, longest response time, resource contention, or loudest complaints. However, are these all the transactions that affect the users service, and in the proper context?

To assess this, we need to view the CICS workload from a terminal user's perspective. User Behavior Graphs (UBG) describe the interaction of the CICS terminal users in a graphical model. The UBG can identify all the transactions related to a business function, uncovering bottlenecks in the users' path through the system. The UBG more accurately describes the workload than a list of transactions. Tuning the UBG will more directly benefit the CICS users than tuning random transactions.

II. THE APPROACH

The CICS systems used in a production environment are very complex. They contain multiple application systems, scores of transactions, hundreds of terminals, and many User Behavior Graphs.

In a paper written by Calzarossa and Trivedi (CMG '86), UBGs are used to build a model for workload characterization and resource utilization. In that paper, each user is represented by their own UBG. In approaching the voluminous data of the production system, to treat user individually becomes impractical. In fact, the Calzarossa and Trivedi solution to this obstacle was to define all the users to be statistically identical when solving their model.

Each CICS system is not a single, massive User Behavior Graph. Under the hood of each CICS system we find multiple UBGs. It is an important distinction to note that a user may operate according to different UBGs at different times, and that the bridge between UBGs must be identified as such. This separation will help in defining the UBG structures of the system.

It follows then that the UBGs to study are those UBGs commonly used within the set of all UBGs.

What does a UBG look like? Calzarossa and Trivedi define the UBGs: each node represents a transaction, each line a probability of transition to another node, and a sojourn time that represents the sum of data entry time, response time, and user think time. The sojourn time is associated with a visit at a node.

Note: reading the Calzarossa and Trivedi paper will give the reader a modeling perspective on this topic, as well as explain in detail their definitions. The method of UBG construction described in sections III and IV of this paper was used for developing UBG charts from actual production system data.

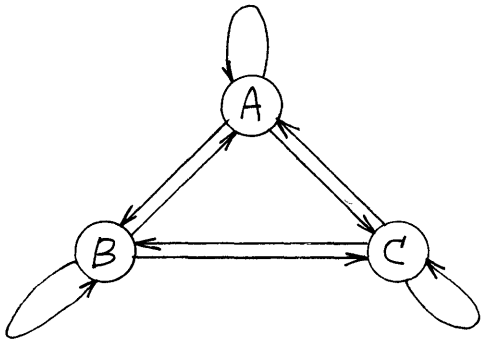


Figure 1 - UBG diagram.

Using a CICS monitor to extract detailed data, the transactions run time and start time are selected. From this data, we can calculate the idle time between successive transactions executed for a terminal using custom BAL programs. Thus, in this papers' representation of the UBGs, the transition has a time associated with it, the idle time. To use the Calzarossa and Trivedi model definition, the run time plus the idle time can be equated to the sojourn time (the time associated with the users visit at the node). It should be noted that the run time does not equal response time. The idle time calculated from the Landmark Monitor's data contains components that should be attributed to response time, such as: MVS, VTAM, CICS overhead and network transmission. When reviewing the Monitor's data, this should be kept in mind.

The limitations of using only a CICS monitor as the source of data prevent a better analysis of the terminal users perspective of the CICS system (ie, we don't have true response time). However, we can build the UBG structures. To do so. we need to transform the detail file into a set of transaction sequence records using the custom BAL programs. Standard SAS procedures are then used to summarize the transaction sequences.

III. TRANSACTION SEQUENCE

A single transaction sequence is composed of two successive transactions executed for a unique terminal. The level of detail needed to generate UBGs does not require anything other than the primary and secondary transaction names of the sequence. Any data associated with the sequence can be carried, and for the UBGs used in this paper the transactions' run times, the idle time between the transactions, and the start time of the first transaction, which is the start time of the sequence, are reported.

The UBG in figure 1 is composed of nine transaction sequences. The probability of transition from each node can be calculated from the nodes' transaction sequences

in which it is the primary transaction. The percentage of the whole UBG that each sequence represents can be calculated. This percentage (the weight) will identify which paths are heavily traveled relative to the UBG.

To develop the global set of all transaction sequences for the CICS system, the detail file must be sequenced by terminal id and transaction start time. The data is then combined into transaction sequence records. The fields selected from each detail record are:

* TATRANS - transaction name
  TASYSID - system identifier
  TADATE - date
  TATERM - terminal identifier
  TASTIME - transaction start time
  TAPTIME - transaction stop time

* Landmark's Monitor field names and definitions.

Other data that are applicable to model construction could be stripped off as well. However, for the UBGs constructed in this paper these data are not needed. (In fact, to build the most skeletal UBG, the primary and secondary transaction names are all that are required.) To demonstrate the analysis technique, some descriptive data are carried. The transaction sequence records in the case studies are composed of the terminal id, sequence start time, primary transaction name and its run time, the secondary transaction name and its run time, and the idle time between the primary and secondary transactions.

To demonstrate the content of a UBG; terminal X executes transaction A, B and C in that order. The transaction sequence records produced are:

Term=X,Primary tran=A,Secondary tran=B
Term=X,Primary tran=B,Secondary tran=C

If no other transactions are executed by X, then the UBG for the user at terminal X is:

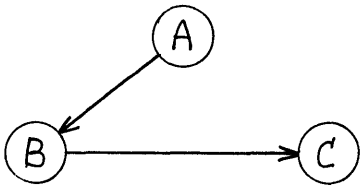


Figure 2 - UBG for X.

In the example UBG in figure 1, there are only three transactions, yet there are nine transaction sequences. If each transaction of a production CICS system could transition to all other transactions, as figure 1 demonstrates, UBG construction would crumble under the weight of computational requirements. In observed systems the actual number of transitions are significantly less. They do not increase geometrically with respect to the number of transactions.

The summarization of the whole set of transaction sequences will give us the components of the CICS systems' UBGs. In the case study appendage, figure 1 shows a transaction summary, and figure 2 is a list of the summarized transaction sequences. Though not an end in themselves, the sequences are informative in that they do demonstrate the interrelationships of the transactions. The UBGs to be constructed will use these transaction sequences as input.

IV. UBG CONSTRUCTION

UBGs are built from two types of transaction sequences. The first is a unary sequence, in which the users' transition is to the same transaction. The second is a binary sequence, in which the user transits to a different transaction.

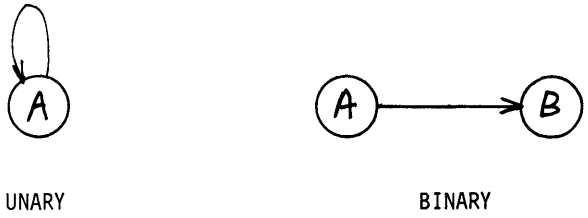


Figure 3 - Sequence types.

UBGs are built from some combination of these two transaction sequence types. The UBGs that are then constructed tend to be of four basic flavors; unary, serial menu, or matrix. In practice, UBGs are combinations of these four.

.Unary: the UBG is composed of a unary transaction sequence only.

Figure 4 - UNARY

.Serial: the transitions tend to flow in a sequential manner through the UBG.

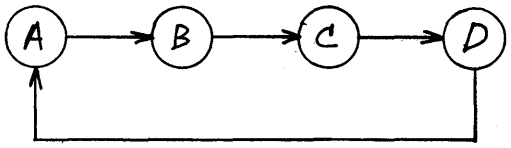


Figure 5 - SERIAL

.Menu: a menu transaction dominates a UBG through which all transitions either enter of leave.

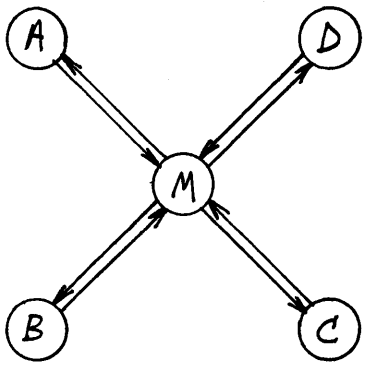


Figure 6 - MENU

.Matrix: each transaction in the UBG can transit to all the other transactions.

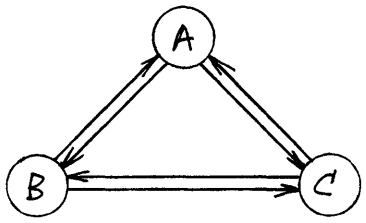


Figure 7 - MATRIX.

.Complex: combinations of the four basic UBG flavors.

When completed, a UBG will describe the interraction amongst the transactions in both diagram and tabular form. (See figures 3 through 7 in the appendage for the case study UBGs.)

We must first select the content of the UBG we are to build. By choosing a "big hitter" transaction as a start, (a big hitter being a major transaction in the system, whether it be selected by: executed most often or longest run time or some other criteria), we can see which UBG it belongs to, and the affect of that transaction on the UBG. Note that a transaction can appear in one UBG only. The sequences associated with the targeted transaction are selected for inclusion in the UBG. The directly related transactions (those that the initial transaction transited to) are then selected. In turn, these transactions are used to select their transaction sequences. This method will select all related transactions, no matter how indirect, including the between-UBG bridge sequences. The tree will continue to grow until we halt selection by some arbitrary condition. The technique best suited to limit the selection process is a transaction naming convention. (For those unlucky souls who have no naming conventions, the transaction selection can be made though table lookups. However, this may require pre-determining the correct transaction mix.) An example will best illustrate this efficient method.

> From the appendage figure 1, we select the transaction executed most on the system, GBLB. What is the context of its use? In the transaction sequence list (figure 2), the GBLB sequences are selected. Then the transactions that GBLB transits to and from are selected and their transaction sequences follow. This will continue indefinitely, so it is here we must limit the iteration of the selection process. Those sequences that include a non-GBLx transaction are presumed to be a bridge to or from another UBG. Hence, we can select the content of the UBG to be only GBLx to GBLx transaction sequences.

Once the appropriate transaction sequences have been selected, to draw the UBG becomes an exercise in connect the dots (er, nodes), using the sequences to direct our connections. The sequences are entered into a table which is then used to calculate the transition probabilities and the weight. The weights are then carried back to the drawing, and marked next to the appropriate transition line. The construction of a single UBG is thus completed, and it is an isolated UBG from the other UBGs in the system.

V. UBG ANALYSIS

GBLx UBG

In figure 3 of the appendage, the GBLx UBG is shown. The table associated with the graph contains the transaction sequences of the UBG. The sequences' count of occurences, the probability of transition from the primary transaction, and the percentage of the whole UBG that the sequence represents are shown.

What does the UBG tell us that the transaction summary list doesn't? We could have guessed that the GBLx transactions were interrelated, but not how, and not with the proper weights given to the paths. Some observations we can make of the GBLx UBG:

- .it is matrix oriented
- .the B to C to B path is most heavily traveled
- .the B to E to B path is second most active
- .the unary activity centered on F and E shows that once arriving at those nodes, the users tend to execute the transaction repeatedly (called an extended visit)
- .the A to B activity is heavier than B to A, possibly reflecting that A is the users' entry into the UBG, and that bridges to other UBGs occur from the other nodes.

Researching further using a batch analysis report shows that transaction B and F share a file. During B's execution, a keyed search is made of the file, and if a browse is needed to locate the correct record, the F transaction is used. Performance gains to this file will improve the run times of B and F. The reduction of this run time and the corresponding response times will enable the users to cycle through the UBG at a faster pace. Assuming a backlogged workload, ie latent demand, we could expect the arrival rate of all the transactions of the UBG to increase. This is easily obtained analytical information that, without the UBG, would have necessitated time consuming effort.

GPSx UBG

Figure 4 of the appendage is the GPSx UBG. Three of the ten most heavily executed transactions (as shown in the transaction summary, figure 1) are in the GPSx UBG. This UBG demonstrates a combination of a menu UBG centered on the A transaction, with heavily used unary sequences at nodes D,C, and T.

How does this differ from the GBLx UBG shown in figure 3 of the appendage? Because of the central nature of A, improvements (or degradation) to transaction A will directly affect the pace of the entire UBG. Changes to the unary oriented transactions will affect the others of the UBG, but not as significantly as the A transaction. However, this is not to say the unary-oriented transactions are less significant in the users perception of performance. Because of those transactions' relative high activity, the D, C and F transactions are significant components of the UBG.

NCxx UBG

Figure 5 of the appendage is the NCxx UBG. It is a menu UBG, centered on the unary sequence O2. Half of all activity in this UBG occurs in the central unary sequence. The many other transaction sequences included in this UBG sum to the remainder, yet individually are insignificant in comparison to the O2 unary sequence. The O2 transaction is involved in 97% of the UBG sequences as either the primary or secondary transaction. The key to users following this UBG is the central transaction.

NAxx UBG

Figure 6 of the appendage is the NAxx UBG. This is a unary UBG, with only the NA77 unary transaction sequence as a component. Users who follow this UBG are affected by the performance of the NA77 transaction only.

GLxx UBG

Figure 7 of the appendage is the GLxx UBG. This is a complex UBG. The 02 transaction acts as a menu to serial legs emanating from it. At the ends of the serial legs are matrix-type structures. Unary sequences occur along the serial and matrix sections, the end result being the complex GLxx UBG.

In analyzing a complex UBG, it is appropriate to split the UBG into sub-UBGs. The 02 transaction can be viewed as the center of a menu UBG, with the 30's, the 20's and the 10's serial UBGs emanating from it. The 32 transaction is heavily used, and is part of a matrix UBG in conjunction with the 34 and 35 transactions. The 32 unary sequence has a large UBG weight, and could be viewed as a unary UBG. The 02 and 32 transactions are end nodes of a serial UBG passing through the 30 and 31 transactions. The serial UBG operates in both directions along the path.

Because of the high activity associated with the 02 and 32 transactions, the serial UBG from 02 to 32 is of secondary importance only to the 32 unary sequence, and the 02 menu sequences.

What does this UBG tell us about the GLxx transactions? The 02 and 32 transactions are important in their own right, and are the keys to the whole UBG. The performance of these two transactions affect the performance of most users of this UBG.

The execution of the UBG occurs in a very structured manner. Path selection restrictions imposed on the users serve to simplify the application to users of the underlying database. If the structure is in place for this reason alone, and not for processing requirements, then expert users (as opposed to novices) will be impeded by this path structure, for they must pass through the intermediary transactions enroute to the desired node. This may or may not be the intention of the design.

VI. CONCLUSION

The construction and analysis of User Behavior Graphs clear the view for the performance analyst. By placing the transactions within the context of their use, we can identify critical paths in the system. In the future, development of software that will do controlled generation UBG models will enable capacity and performance studies of the CICS workload more directly based on the terminal users interaction with the CICS system.

VII. ACKNOWLEDGMENTS

Thanks to Ted Osmer, Melvin Swanson, and Bill Yarnall for their constructive editing and advice, and most especially to Ted for his encouragement.

REFERENCES

1) Calzarossa, M. and Trivedi, K.
Performance Analysis Using User Behavior Graphs.
CMG 1986 pp.394-397

2) Landmark Systems Corporation, The Monitor for CICS, Users Manual Version 6.2, 1986

3) SAS Institute Inc., SAS Users Guide: Basics, Version 5 Edition, 1985

APPENDIX

Figure	Description
1	Transaction Summary
2	Transaction Sequences
3	GBLx User Behavior Graph
4	GPSx User Behavior Graph
5	NCxx User Behavior Graph
6	NAxx User Behavior Graph
7	GLxx User Behavior Graph

FIGURE 1: TRANSACTION SUMMARY 1
11:09 MONDAY, AUGUST 31, 1987

RUN DATE=870326

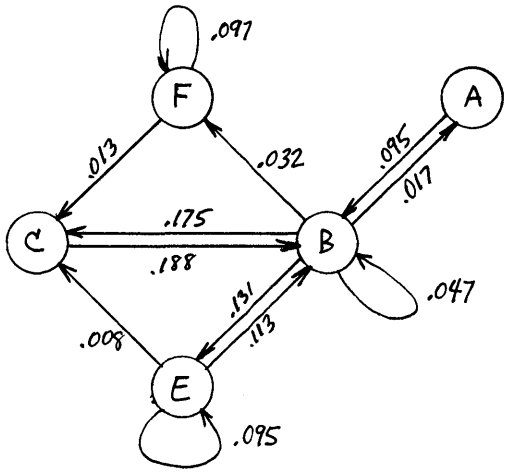
TRANSACTION	COUNT	AVERAGE RUN TIME	RUN DATE
GBLB	3005	1.070	870326
NC02	2920	0.212	870326
GC80	1888	0.177	870326
NA77	1612	0.928	870326
GBLE	1515	0.701	870326
GBLC	1366	0.042	870326
GPSA	1212	0.275	870326
GPSD	925	0.274	870326
GPSC	837	0.440	870326
GBLF	777	3.316	870326
GLIP	702	4.042	870326
GBLA	672	0.037	870326
GL02	502	0.171	870326
GPSF	481	0.126	870326
NB02	474	0.253	870326
GL32	414	0.427	870326
GL01	351	0.353	870326
NC84	338	0.120	870326
GPSG	276	0.169	870326
GL30	266	0.366	870326
OFF	261	0.040	870326
GL10	255	0.204	870326
NC30	250	0.578	870326

FIGURE 2: TRANSACTION SEQUENCES 1
11:06 MONDAY, AUGUST 31, 1987

RUN DATE=870326

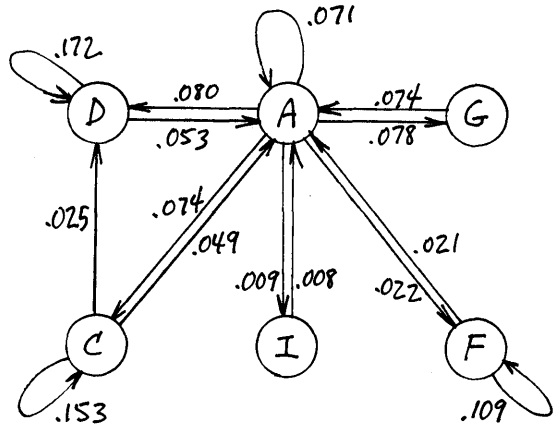
PRIMARY TRANSACTION	SECONDARY TRANSACTION	COUNT	PRIMARY TRANSACTION RUN TIME	IDLE PERIOD AVERAGE	SECONDARY TRANSACTION RUN TIME
NC02	NC02	1706	0.209	49.31	0.204
NA77	NA77	1571	0.950	34.05	0.935
GC80	GC80	1399	0.156	70.96	0.161
GBLC	GBLB	1269	0.042	66.86	1.060
GBLB	GBLC	1181	0.970	116.64	0.044
GBLB	GBLE	886	0.612	28.10	0.576
GBLE	GBLB	763	0.565	79.70	0.708
GLIP	GLIP	702	4.042	6.19	4.016
GBLE	GBLE	642	0.874	25.29	0.879
GBLA	GBLB	641	0.037	28.70	1.637
GPSD	GPSD	633	0.203	17.22	0.300
GBLF	GBLF	567	3.679	10.14	3.399
GPSC	GPSC	562	0.366	95.50	0.441
NB02	NB02	441	0.230	28.33	0.254
GPSF	GPSF	400	0.128	59.23	0.125
NC84	NC84	330	0.118	24.24	0.115
GBLB	GBLB	317	0.338	46.32	0.798
GPSD	GPSA	292	0.427	15.43	0.365
GPSA	GPSG	288	0.371	18.05	0.170
GPSA	GPSC	273	0.215	87.13	0.440
GPSG	GPSA	273	0.169	196.69	0.191
GPSA	GPSA	261	0.221	95.85	0.272
GL01	GL02	254	0.439	46.86	0.176
NC02	NC30	240	0.177	22.96	0.551
NC30	NC02	240	0.590	52.23	0.196
GL32	GL32	227	0.455	35.79	0.463
GOLD	GL01	221	0.220	22.84	0.379
GBLB	GBLF	216	4.064	21.46	3.052
GPSA	GPSD	194	0.403	23.49	0.191

Figure 3 - GBLx UBG



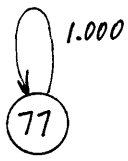
Primary Transaction	Secondary Transaction	Sequence Occurence Count	Probability of Transition	Transaction Sequence Weight
A	B	641	1.000	.095
B	A	116	.042	.017
B	B	317	.117	.047
B	C	1181	.435	.175
B	E	886	.326	.131
B	F	216	.080	.032
C	B	1269	1.000	.188
E	B	763	.522	.113
E	C	57	.039	.008
E	E	642	.439	.095
F	C	88	.134	.013
F	F	567	.866	.097

Figure 4 - GPSx UBG



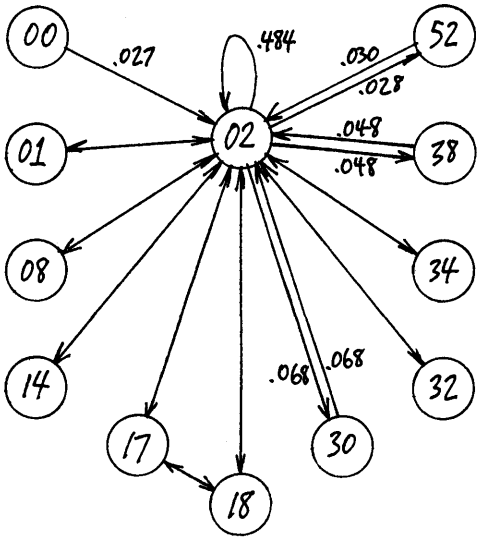
Primary Transaction	Secondary Transaction	Sequence Occurence Count	Probability of Transition	Transaction Sequence Weight
A	A	261	.231	.071
A	C	273	.241	.074
A	D	194	.172	.053
A	F	82	.073	.022
A	G	288	.255	.078
A	I	33	.029	.009
C	A	179	.215	.049
C	C	562	.675	.153
C	D	92	.110	.025
D	A	292	.316	.080
D	D	633	.684	.172
F	A	78	.163	.021
F	F	400	.837	.109
G	A	273	1.000	.074
I	A	30	1.000	.008

Figure 6 - NAxx UBG



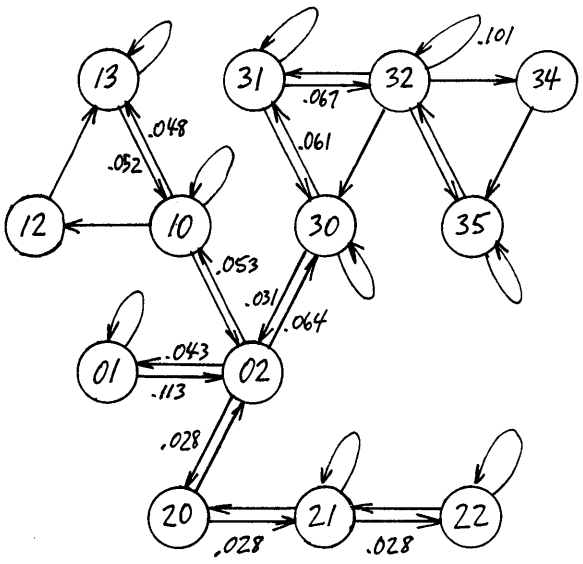
Primary Transaction	Secondary Transaction	Sequence Occurence Count	Probability of Transition	Transaction Sequence Weight
77	77	1571	1.000	1.000

Figure 5 - NCxx UBG



Primary Transaction	Secondary Transaction	Sequence Occurence Count	Probability of Transition	Transaction Sequence Weight
00	02	96	1.000	.027
01	02	45	1.000	.013
02	01	45	.017	.013
02	02	1702	.669	.484
02	08	57	.022	.016
02	14	41	.016	.012
02	17	55	.022	.016
02	18	30	.012	.009
02	30	240	.094	.068
02	32	31	.012	.009
02	34	69	.027	.020
02	38	169	.066	.048
02	52	104	.041	.030
08	02	45	1.000	.016
14	02	41	1.000	.012
17	02	33	.532	.009
17	18	29	.468	.008
18	17	37	.561	.009
18	18	29	.439	.008
30	02	240	1.000	.068
32	02	30	1.000	.009
34	02	70	1.000	.020
38	02	168	1.000	.048
52	02	100	1.000	.028

Figure 7 - GLxx UBG



Primary Transaction	Secondary Transaction	Sequence Occurence Count	Probability of Transition	Transaction Sequence Weight
01	01	29	.102	.013
01	02	254	.898	.113
02	01	96	.227	.043
02	10	120	.284	.053
02	20	63	.149	.028
02	30	144	.340	.064
10	02	83	.384	.037
10	10	14	.065	.006
10	12	11	.051	.005
10	13	108	.500	.048
12	13	11	1.000	.005
13	10	118	.776	.052
13	13	34	.224	.015
20	02	33	.344	.015
20	21	63	.656	.028
21	20	35	.276	.016
21	21	29	.228	.013
21	22	63	.496	.028
22	21	39	.549	.017
22	22	32	.451	.014
30	02	71	.292	.031
30	30	35	.144	.016
30	31	137	.564	.061
31	30	27	.144	.012
31	31	10	.053	.004
31	32	150	.802	.067
32	30	57	.151	.025
32	31	56	.149	.025
32	32	227	.602	.101
32	34	27	.072	.012
32	35	10	.027	.004
34	35	25	1.000	.011
35	32	32	.727	.014
35	35	12	.273	.005

VSAM SPECIFICATION AND TUNING:
OPTIMIZATION OF VSAM PERFORMANCE IN BATCH AND CICS ENVIRONMENTS

Frank Bereznay
Automobile Club of Southern California

ABSTRACT

Proper specification of a VSAM data set can have a major impact on the amount of DASD space needed to store the data and the resources required to process it. This paper summarizes experiences gained from VSAM tuning efforts. The paper begins with an overview of the structure of a VSAM Key Sequenced Data Set (KSDS) and a discussion of 3380 DASD architecture as it relates to track utilization percentages for the various control interval sizes. A DEFINE CLUSTER statement will then be used to review specification options and examine the interdependence between the size of the data control interval, index control interval, control area and the length of the retrieval key. Buffer management follows, showing how proper buffer specification can reduce the number of data transfers needed to process a record. This section includes techniques on how to calculate the number of index buffers in a VSAM data set and how locally shared resource (LSR) pools function. The final topic of the paper addresses update processing which includes an explanation of the potential overhead associated with it and how freespace can be included in the data set to reduce or eliminate this overhead.

INTRODUCTION

VSAM, the virtual storage access method, is widely used in system and application areas and yet, despite this widespread usage, there tends to be a lack of understanding of how these data sets should be specified and tuned. It has been observed that a common technique to specify a VSAM data set is to locate a DEFINE CLUSTER statement that exists for some other data set and then change the minimum number of parameters needed to make it work for the new data set. While this technique typically generates a return code of zero from IDCAMS, the data set produced may not be what is really wanted or needed. The VSAM specification/tuning task is even more critical in a CICS environment where many (frequently over 100) data sets are allocated and items such as buffer sharing and string numbers can have a major impact on response time and resource usage.

This paper summarizes the experiences gained during VSAM tuning projects. It will identify key specification areas and how these items can have a significant effect of the amount of DASD space required to store the data and resources required to process a record. In an online environment, attention to these details can translate into a dramatic reduction in response time. In batch environments it can represent major reductions in run times. In both environments proper specification and tuning can reduce the resources required to process records.

OVERVIEW

A KSDS (Key Sequenced Data Set) is a type of VSAM data set that supports sequential and direct processing. When direct processing is performed records are retrieved and written based upon the value of a key field that is part of the data record. Sequential processing is performed by retrieving or writing records in ascending sequence based upon the value of the key field.

There are three catalog entries associated with a VSAM KSDS data set. These entries are; (1) cluster entry, (2) data component entry, and (3) index component entry. The cluster entry contains the name used to reference the data set and establishes the connection or association between the cluster name and the two component names which make up the data set. The data component entry locates the part of the data set that contains the user data. The index component, which actually consists of two parts, is a collection of retrieval keys and pointers which are used to allow keyed retrieval, updating, and insertion of records within the data set. It is organized in a hierarchical manner, normally with three levels. The bottom level of the index component is called the sequence set and all upper levels are collectively called the index set.

The control interval represents the amount of data that is transferred to and/or from disk as part of an I/O operation. It is NOT necessarily the physical block size for the data set. In many cases the actual physical record size is smaller than the control interval specified. The control interval can also be thought ot as a collection of records that are pointed to by an entry in a sequence set record of the index component.

A control area is a collection of control intervals, usually a cylinders worth. There is a one-to-one relationship between the number of control areas in a VSAM KSDS and the number of sequence set records in the index component. This one-to-one relationship implies that a sequence set record must be large enough to hold a high key entry for each control interval in the control area. If you assume a 4K data control interval size and a full cylinder control area there will be 150 control intervals in the control area (on a 3380 device). The sequence set record in this case must be large enough to store a key value and pointer for each of the 150 control intervals within the control area.

The retrieval key, interestingly enough, is key to the proper specification of a VSAM KSDS data set. This is the character string used to locate records for direct processing. VSAM is very sensitive to the size and compressability of these keys since they must be stored in the index component. It follows that the length of the retrieval key is a factor to be considered when choosing the control interval size for the index component and the size of the control area for the data component.

3380 OVERVIEW

The 3380 DASD device is not a byte addressable storage medium, data is stored in multiples of 32 bytes plus some overhead for system usage. For VSAM data sets this implies that the percentage of the track that is used for data will differ based upon the size of the control interval that is chosen. This difference can be quite significant with the track utilization percentage ranging from 50% to 85% of total track capacity. Figure 01 is a chart of control interval sizes and the amount of data they store on a track under DFP version 1. Track utilization data will change with DFP version

2 because physical record sizes will match the control interval size.

Choosing the proper control interval size depends upon the intended use of the data set. When it is used in batch applications the only consideration is storage efficiency. However, when the data set is used in a CICS environment the control interval size establishes what buffer pools will be used for the data set. Unless the data set is exceptionally large, control interval sizes should be standardized to improve CICS buffer management. They should also be kept on the smaller side to minimize data transfer time and buffer usage for direct processing.

As a general set of guidelines, the following control interval sizes should be used for VSAM KSDS's in a CICS environment for small to medium sized (less than 2K) logical records. The 80/20 rule applies here. These CI sizes should be used for the 80 percent of the data sets that represent 20 percent of the storage allocated. In most cases the better track utilization of the data CI size will outweigh a poor fit based upon logical record length. The other factor to consider is the length of the retrieval key which will be discussed later.

Primary Access Method	Data CI Size	Index CI Size
Direct	4096	2048
Sequential	8192	1024

FIGURE 01

3380 Track Utilization Data For
Various VSAM Control Interval Sizes

CI Size	CI Per CA	User Data	% Utilized
1024	465	476,160	78
1536	230	353,280	58
2048	270	552,960	90
2560	138	353,280	58
3072	155	476,160	78
3584	98	351,232	57
4096	150	614,400	100
5632	62	349,184	57
6144	90	552,960	90
6656	53	352,768	57
7168	66	473,088	77
7680	46	353,280	58
8192	75	614,400	100
10240	54	552,960	90
12288	50	614,400	100
14336	38	544,768	89
16384	37	602,208	99
18432	30	522,960	85
20480	30	614,400	100
22528	24	540,672	88
26624	20	532,480	87
28672	21	602,112	98
30720	18	552,960	90
32768	18	589,824	96

NOTE: These track utilization calculations are for a DFP version 1 environment.

The track utilization percentages are based upon 614,400 bytes which is the maximum amount of user data that can be stored on a track.

SPECIFICATION

A DEFINE CLUSTER statement is prepared and executed by IDCAMS to allocate a VSAM data set. The following is an example of a DEFINE CLUSTER statement for a KSDS:

```
DEFINE CLUSTER (NAME(POOR.EXAMPLE.KSDS)          -
                FREESPACE (0 10)                 -
                IMBED                            -
                KEYS(0 12)                       -
                RECORDSIZE(225 225)              -
                SHAREOPTIONS(2 3)                -
                VOLUME(PON832))                  -
          DATA (NAME(POOR.EXAMPLE.KSDS.DATA)     -
                CONTROLINTERVALSIZE(4096)        -
                CYLINDERS(30 5))                 -
         INDEX (NAME(POOR.EXAMPLE.KSDS.INDEX)    -
                CONTROLINTERVALSIZE(512)         -
                TRACK(1 1))
```

VSAM specification options that can affect performance, and are independent of the physical characteristics of the data set, include selection of control interval size, sizing of the control area and use of IMBED to reduce seek activity.

SIZING THE CONTROL AREA

Large retrieval keys can waste tremendous amounts of disk space unless control interval sizes and possible control area sizes are adjusted to accommodate for the large key. This is a passive specification problem because no error or warning message is issued when a control area is partially loaded because a sequence set record filled up.

The control area is a collection of control intervals. The size is usually one cylinder, however it can be as small as one track. Each control area is assigned one sequence set record to hold a compressed high retrieval key for each control interval in the control area. If the retrieval key is large, or it does not compress, the sequence set record will fill up before the control area is loaded. When this occurs the loading of that control area is stopped and VSAM moves on to the next control area to continue the loading of the data set. The remainder of the first control area is now dead space or unusable.

The following factors should be considered when reviewing a control area specification:

- Size of the control area - This is determined by the space allocation values for the data component. It is the smaller of the primary allocation, secondary allocation and one cylinder.
- Data control interval size - This determines the number of control intervals within the control area.
- Index control interval size - This determines the size of the sequence set record.
- Length and compressability of retrieval key - The size of the compressed retrieval key times the number of control intervals in the control area plus some control fields determines how big the sequence set record must be to fully use the control area.

Using the above DEFINE CLUSTER as an example we find that:

- The control area is one cylinder. This is the value used since the primary and secondary

allocations are greater than one cylinder. In order to define sub cylinder control areas the space allocation for the cluster must be in tracks with either the primary or secondary allocation being the desired control area size.

- There are 140 control intervals in the control area. A 4096 size results in 10 control intervals per track. Since IMBED is chosen there will be 14 tracks of data, (the 15th track is used to store a replicated sequence set record) which accounts for the 140 control intervals.

- The sequence set record is 512 bytes in length. The IMBED option DOES NOT increase the size of the sequence set record. It causes the 512 byte record to be repeated as often as possible on a single track.

- The retrieval key is 12 bytes in length. It is very difficult to determine the compressability of a retrieval key without studying the data itself. Generally speaking larger keys compress better than smaller keys and old data sets have keys that compress better than new data sets. The standard assumption is a 50% compression factor.

The following is a formula to aid in calculating the capacity of a sequence set record assuming a 50% compression of the retrieval key.

```
# of Entries = (CI SIZE - 31)/((Key Length/2) + 5)
             # of Entries = 43.7
```

In this example the sequence record would fill up after about 43 to 44 entries (or control intervals). Since the control area contains 140 control intervals the remaining 95 to 96 would be unusable. Roughly 68% of the cylinder would be wasted.

If the index control interval size were increased to 2048 the capacity of the sequence set record would be about 183 entries. This would allow the sequence set record to support a full control area's worth of control intervals before filling up.

If the index control interval size could not be increased then the control area should be made sub-cylinder to allow for full cylinder utilization. This would be done by making the space allocation in tracks and making the secondary allocation 4 tracks. When allocating sub-cylinder control areas and IMBED is used, the allocation must not include the track used for the imbedded sequence set record. In this example the 4 track control area would be combined with one track for the sequence set record making the total control area 5 tracks. This would divide evenly into the 15 tracks of a cylinder allowing 3 control areas per cylinder.

MISCELLANEOUS SPECIFICATION NOTES

The IMBED option causes the first track of a control area to be used to store the sequence set record assigned to that control area. The sequence set record is replicated around the track to minimize rotational delay. By placing the sequence set record and data on the same cylinder a single seek will normally satisfy both record retrievals. Using the IMBED option with a full cylinder control area will reduce the data storage by 1/15 or roughly 7%. This figure will be higher for sub-cylinder control areas.

This option is recommended because it can improve performance and because it reduces the number of control intervals in a full cylinder control area. Assuming a 4K data control interval on a 3380 device a full cylinder control area consists of 150 control intervals. If IMBED is specified the number is reduced to 140. Reducing the control area by 10 control intervals is not a major reduction but it can help keep a sequence set record from filling up.

Keep data and index components on the same volume. It can be argued that splitting the components reduces head contention and this can be true. However, a better solution is to keep the index set resident in memory and use IMBED to allow a single seek to serve two purposes. Splitting the components increases the complexity of data recovery in the event a volume fails. A full volume restore of half a VSAM KSDS may render the entire data set useless.

CONTROL AREA SIZING EXAMPLE

A set of VSAM specification standards were developed for a CICS environment where VSCR (Virtual Storage Constraint Relief) was a concern. The standards included a 2K data control interval size and a 1K index control interval size with IMBED as the default. Around 225 data sets were defined using these standards and they occupied 7 single density 3380 disk volumes. The 2K/1K control interval sizes would require sub-cylinder control areas for retrieval keys as small as 2 bytes. The average retrieval key length for these data sets was over 20 bytes and full cylinder control areas were used.

The problem was initially identified by the DASD management group, when they compared the user data being backed up to the amount of disk space allocated to the DASD pool. The VSAM data sets represented 1.1 giga-bytes of user data and 3.1 giga-bytes of DASD was being allocated to it. This led to the development of a storage efficiency ratio; user data divided by storage allocated.

The initial efficiency ratio for these VSAM data sets was 31%. For every 100 cylinders allocated only 31 contained user data. The remainder were used for system overhead purposes or dead space. During the next three months these VSAM data sets were respecified using sub-cylinder control areas, typically 3 or 5 tracks. At the end of the project the storage efficiency ratio increased to 49%. This represents a 58% increase in track utilization or a reduction of over 2 disk volumes with no reduction in user data.

Some additional work went into analyzing the storage efficiency ratio. While 49% was better than 31% it still seemed pretty low. Under ideal conditions the ratio could get into the mid 70%'s. This required a perfect fit of the logical record in the control interval and no specification of freespace in the data set for update purposes. For data sets that were updated the best storage efficiency ratio was in the low 60%'s. After the sizing issue was corrected, logical record fit and freespace specification tended to have the greatest impact on the storage efficiency ratio.

BUFFER MANAGEMENT

The objective of buffer management is to keep frequently referenced records in memory to improve response time for direct processing and reduce resource consumption. If minimum buffering is used, the average VSAM data set will require four data transfers to retrieve a record. There will be a separate record retrieval for each level in the index component and one record retrieval for the data control interval.

Buffers can be allocated to keep the index set, usually 4 to 10 records, memory resident. This can cut the time for the I/O process in half. Furthermore, the processor time associated with performing the I/O is eliminated. In this situation you can improve response time and at the same time reduce CPU usage by using a small amount of virtual and real storage.

CALCULATING THE SIZE OF THE INDEX SET

The index component consists of fixed length records, with the record length being the control interval size. The easiest way to calculate the number of records in this component is to divide the high used RBA (Relative Byte Address) by the control interval size. The record count field in the LISTCAT listing may not be accurate. Depending upon how the cluster was defined, some additional calculations may be needed to determine the size of the index set.

1) IMBED and NOREPLICATE - Take the high used RBA for the X'00' extent type of the index component and divide it by the index control interval size. This should divide evenly and represent the number of records in the index set. When IMBED is specified the sequence set records are physically located with the data and are not part of the X'00' allocation.

2) IMBED and REPLICATE - Take the high used RBA for the X'00' extent type of the index component and divide it by the index control interval size. VSAM adjusts the RBA address to compensate for the record replication. This should divide evenly and represent the number of records in the index set.

3) NOIMBED and NOREPLICATE - The X'00' extent will contain sequence set records in the allocation so they must be removed from the calculation. To do this you must take the high used RBA for the data component and divide it by the control interval size and then by the number of control intervals in a control area. This result is the number of active control areas in the data set, i.e. the number of sequence set records. Then calculate the index set as in example #1 and subtract the number of sequence set records from it. The result is the number of records in the index set.

4) NOIMBED and REPLICATE - Similar to number 3 the index allocation contains sequence set records. Calculate the number of active control areas as in number 3. Then calculate the size of the index component as in number 2. Finally subtract the number of active control areas from the size of the index component and the result is the size of the index set.

CALCULATING BUFFER REQUIREMENTS FOR BATCH PROCESSING

The index buffer allocation for direct access batch processing will be the size of the index set plus 2 (one for the sequence set record and one to cover a CA split). The data buffer allocation should be 2 (one for data and one for CA split). When allocating buffers for sequential processing allocate two index biggers and approximately 14 data buffers. It is recommended that these buffer specification be made in the JCL, not as part of the definition of the data set or as part of the open.

CALCULATING BUFFER REQUIREMENTS FOR CICS PROCESSING

The CICS environment provides two techniques for VSAM buffer management: (1) non-shared buffer allocation and (2) locally shared buffer pools. The non-shared technique established a separate set of file buffers for each data set and the shared technique allows a group of data sets to use a common pool of buffers. There are advantages and disadvantages to each technique but generally share buffer pools are recommended if possible.

CICS allows multiple tasks to concurrently access a single data set. This multi-tasking process places some additional demands on the size of the buffer pool. Typically you will need one additional index record for each concurrent user. The parameter that control the number of concurrent users is called the string number (STRNO) and is specified in the FCT for the data set.

NON-SHARED RESOURCE (NSR) BUFFER POOLS. For direct processing use the size of the index set plus 1 (CA split) plus the string number. If the string number is 5 it implies 5 users may concurrently process the data set and each user may come up with a unique sequence set record. To keep these sequence records from overlaying the index set, the buffer number is increased accordingly. These values will be placed in the CICS FCT table entry for the data set.

LOCALLY SHARED RESOURCE (LSR) POOLS. As the name implies this technique uses a common set of buffer pools for multiple data sets. As a default, the size of the pool is determined by the index and data buffer specifications for the data sets that are assigned to it. CICS will sum the values and create buffer pools 1/2 the number summed. The size of the buffer pools can be further adjusted with a macro at the end the FCT table.

Shared buffer pools have two main advantages: (1) they use less virtual and real storage because multiple data sets share the same buffer pools, and (2) it is possible to read all records associated with a record request from in memory buffers. When using non shared buffer management only the index set will be read from in memory buffers. With shared buffer pools the entire index component and the data control interval will be read from in memory buffers if present.

When data sets are assigned to an LSR buffer pool the index and data buffer specifications will not limit the number of buffers the data set may use. As many buffers as needed will be allocated from the pool. The retention of these buffers in memory will be determined by the size of the pool and the frequency of reference of the records. However, VSAM will recognize the string number specified to limit the number of concurrent users of the data set. This allows the string number parameter to be used as a limiting factor to keep a high use data set from overwhelming a buffer pool.

The buffer pools are managed on a least recently used (LRU) basis. When the buffer pool fills up the record that hasn't been referenced for the longest time will be purged. To evaluate the success of a buffer pool a hit ratio is used. This is the ratio of records read out of memory divided by total record requests. For index buffer pools this percentage should be in the high 80's to low 90's. Data buffers are normally in the high 40's to mid 60's. CICS shutdown statistics can be used to develop this ratio.

The size of the buffer pools should be adjusted until the proper hit ratios are achieved. Making the pools excessively large will drive up processor utilization

because the pools are frequently serially scanned. If they are left too small there will be little or no successful look aside processing and transactions may be delayed waiting for buffer availability.

When data sets use this buffer technique it is very important to use different control interval sizes for data and index components. Records are assigned to buffer pools based upon the control interval size, not their use. Data and index records will end up in the same buffer pool if they have a common control interval size. In this situation the data buffers will tend to crowd out the index records because of their greater number.

CICS BUFFER MANAGEMENT EXAMPLE

An application owning region in a CICS MRO complex was experiencing frequent program compressions (greater than two a minute), which was assumed to the source of poor service. To eliminate this problem the MRO complex was expanded and half of the workload was moved into a separate region. After the change was implemented program compressions were reduced, however, response time improved only slightly averaging 3.5 seconds. The bottleneck became string waits on high usage VSAM data sets. A group of 5 VSAM data sets were getting between 2,000 and 3,000 string waits during a 10 hour day. OMEGAMON CICS showed that usage and queueing for these data sets was the major wait reason for the high response time.

These 5 VSAM data sets were reviewed and their entries in the FCT were changed to keep the entire index set memory resident and at the same time the string number was increased. It was calculated that the increased buffer specifications required 96K additional memory. This set of changes had a dramatic affect on all measures of the CICS workload.

The average response time dropped from 3.5 seconds to .5 seconds. At the same time the CPU usage of the region dropped from an average of 28 minutes per hour to 12 minutes per hour a reduction of 57%. Finally the transaction volume increased 13%, from an hourly average of 15,500 to 17,500.

The reduction in processing time was partially due to the fact that the total I/O volume for the region dropped 200K, from 1.4M to 1.2M. That can only partially account for the 57% reduction. We have concluded the remainder is the overhead associated with maintaining the transactions for it's active life. Before the changes, CICS had to manage the transaction for an average of 3.5 seconds and after the change it dropped to .5 seconds. During the life of the transaction it would be on the active and suspend chains and a set of control blocks and user data storage would be associated with it. Maintaining these resources for the extra time, 3 seconds, is assumed to account for the CPU usage that was recovered by reducing the response time.

These improvements were achieved without using LSR buffer pools for the data sets. The use of the LSR buffer pools requires that a task cannot have multiple active I/O operations against a data set. The most common case of this is where a browse is started to locate a record and a get for update is issued before the browse is ended. This programming technique will work in a NSR environment but will create a "deadly embrace" in a LSR environment. The application logic needs to be checked and tested for this condition before LSR buffer pools can be used. A resource was not available to do this so the decision was made to stay with NSR buffer allocations for the high use data sets.

UPDATE PROCESSING

DEFINITION OF SPLIT PROCESSING

When records are added, and when they are updated if their record length increases, VSAM may be required to divide a control interval in half to provide space for the new or updated record. This is called split processing and it can occur at the control interval and control area level. The amount of split processing that will occur is determined by the pattern of update activity and the amount of freespace specified when the data set was defined.

Generally speaking, less is better when it comes to split processing. The main concern here is the overhead that VSAM incurs to perform the split processing. Control interval splits, dividing a single control interval, represent slight to moderate overhead. Control area splits, dividing an entire control area in half, represents significant to major overhead and should be kept to a minimum. Additionally, after a control area split, logically ascending data is no longer physically adjacent. This slows down sequential activity and tends to increase seek times for direct processing if it is clustered.

Assume that a VSAM KSDS was defined with no control interval or control area freespace. During the initial load VSAM will place as many records as possible in each control interval and use every control interval in a control area (assuming the sequence record is large enough). In this situation, the first record that is added to the data set will cause a control interval and a control area split.

A control area split will create a new control area and move 50% of the control intervals of the first control area into the new one. Assuming the 4K data control interval size and full cylinder control areas, this implies that 75 control intervals must be moved from one control area to another. During this process the index component is also modified to reflect the new location for the control intervals. Needless to say, this is a significant amount of data movement and can be the cause for delays in response time or batch throughput.

Once the control area split is completed, VSAM can continue with the control interval split by retrieving one of the empty control intervals and putting half of the original control interval in it to allow for the new record to be added.

FREESPACE SPECIFICATION

Freespace can be specified as a DEFINE CLUSTER option to provide pockets of empty space to allow updates to occur without having to do splits. It can be specified at the control interval and the control area level. Freespace is specified as a percentage and it represents the amount of space VSAM reserves during the initial load of the data set. Freespace percentages are not used during split processing, split processing always divides the control interval or control area in half. Specifying 25% control interval freespace with a 4K data control interval means 1K of the control interval is reserved. Making the same specification at the control area level implies that 38 of the 150 control intervals in the control area would be reserved.

The proper type of freespace, i.e. control interval or control area, depends upon the update patterns for the data set. If the update patterns are clustered then control area freespace is most effective. If the update activity is uniform across the entire data set then control interval freespace is a better choice.

With clustered update activity you let the first update split the control interval. Subsequent activity will be in the same area so there is a good probability that the original control interval, or the new one, will be used again. Remember that split processing always splits the control interval in half so there should be ample room for subsequent updates. In this case, specifying control area freespace keeps the control area splits from occurring.

When the update activity is uniform you assume any given control interval will only be updated once or twice. In this case you specify freespace at the control interval level and hopefully no split processing will occur.

Typically speaking, most update patterns seem to be clustered. Therefore, when you create a new KSDS that is to be updated, specify between 10% to 15% control area freespace and watch it for a couple of weeks or so to determine the update activity.

When KSDS consists of fixed length records, or if it consists of variable length records whose length does not change during update processing, you can use the IDCAMS listcat report to determine the pattern of update activity. Create the data set with only control area freespace and then compare the number of inserted records to the number of control interval splits. If the update activity is clustered you will get more inserted records than there are control interval splits. This implies that the split control intervals are being used for more than one record insertation. If, on the other hand, the number of inserted records is equal or close to the number or control interval splits you can assume the update patterns are uniform. In that case the control area freespace should be lowered and control interval freespace should be specified.

REFERENCES

[1] Using IMBED and REPLICATE in VSAM (Parts 1 & 2), Candle Computer Report: Volume 6 Number 5 & 6, March 1984.

[2] Optimizing VSAM Index Buffers (Part 1 thru 3), Candle Computer Report: Volume 6 Number 7 thru 9, April & May 1984.

[3] VSAM-ASSIST Users Guide, Softworks Corporation 1987.

PRODUCTION ALLOCATION GUIDELINES

KSDS Definition

I. Standardize the control interval sizes
   - Direct processing 4K Data and 2K Index
   - Sequential processing 8K Data and 1K Index

II. Use full cylinder control areas
   - Review length of retrieval key

III. Use IMBED option

IV. Specifiy FREESPACE based upon type of activity
   - Control area for clustered update activity
   - Control interval for uniform update activity

Buffer Specification

I. Always keep index set memory resident

II. Use CICS LSR buffer pools whenever possible

IN PURSUIT OF DASD DATA PATH ALTERNATIVES

Ben Duhl
Storage Technology Corporation

ABSTRACT

Channel data transfer speeds and current data path connection architecture in the large IBM system environment may soon include faster transfer rates and additional pathing. This paper addresses the impact of these changes on subsystem performance via the use of simulation models and data processing workload profiles. Two key issues are presented: the impact of faster channels vs. conventional channels and enhanced subsystem porting architecture vs. current data pathing.

To attain knowledge learn things every day.
To attain wisdom remove things every day.
Lao-tse

BACKWARD

Is the adventure of improving DASD subsystem performance over? In the 1970s disk device service times saw reductions of 50% when the 3350 class DASDs were introduced. By 1980 the thin film head 3380 class DASDs boosted data transmission rates by a factor of 2.5 in addition to reducing mechanical seek times. The double capacity DASD devices came into service in 1985, retaining similar device service time characteristics, emphasizing the cost per megabyte and reduced floorspace requirements. In previous papers [1; 3] the performance issues of single capacity (1X) and double capacity (2X) device expectations were addressed. Highlights [1] included an expected reduction in maximum sustained throughput of about one-third with 2X devices configured for equal disk platter capacity as a 1X subsystem, and poorer IO response times with cached controllers servicing 2X configurations with unbalanced or skewed IO activity.

The recent introduction of yet faster mainframes from multiple vendors continues to emphasize the need for increasing parallelism in data processing activity, particularly for commercial as opposed to scientific applications. 3090 class mainframes increased the number of channels, for instance, from a maximum of 96 up to 128. Such hardware enhancements accommodate system parallelism growth requirements for online and interactive environments. But channel speed, for the time being, remains the same at 3 MB/sec. Speculation abounds about faster channel speeds for rotating DASD, either in native mode or only with cached controllers servicing IO previously staged into the controller memory. This paper addresses performance expectations and limitations associated with faster data paths and channels.

Increasing demand for current and new online software applications generally results in the need for more magnetic storage capacity. Enlarging disk spindle capacities has proven to be an economical alternative to acquiring more 1X spindles as part of the disk subsystem configuration. With increased data capacity residing on the same or lesser number of disk actuators, the challenge from a product performance viewpoint is to control IO wait times due to deferred requests caused by device busy or data path contention. Future disk subsystems will likely exploit faster data transfer speeds as well as more sophisticated alternate data paths between CPU mainframes and IO storage devices. Our intent is to evaluate such design options using a software performance model.

THE MODEL: ROUNDING UP THE USUAL SUSPECTS

To expand on a quote from Mark Twain, "rumors of the death of simulation models have been greatly exaggerated." Monte Carlo simulation models are very effective for evaluating new designs and functions on IO subsystems. A discrete event simulation model is structured in the IBM GPSS language [2] to reflect a variety of currently existing and hypothetically possible disk storage devices. The model permits us to represent not only the service time components of the rotating disk devices but also the various pathing components, including the control modules or heads of string, the directors, and the channels. We will vary key device characteristics as well as IO workloads which represent batch, online, and backup and restore operations. Additionally we will examine future designs as they relate to cached controllers and solid state disk devices.

A Monte Carlo design assures the stochastic effect on IO workload arrival rates, thus adding a measure of reality and a reflection of robustness to the observed performance results. Random number generators create peaks and valleys on IO requests which are independent of one another and conform to the Poisson distribution curve. In the open loop mode, we approximate the classical M/M/1 queueing environment for the IO devices while varying server characteristics in comparative simulation experiments. Dual ports (aka dynamic level selection, or DLS) and quad ports are more accurately represented as multiservers, perhaps M/M/2 or M/M/4, subject to constraints and connectivity with control modules and channels. Simulation models are especially well suited for representing current and future data path design alternatives without making progressively more complex mathematical assumptions a priori, as is the case with some analytical models.

In the closed loop mode, we are interested in capturing theoretical (and also unattainable) maximum sustained throughputs for a given disk subsystem configuration and a given IO workload. The closed loop mode uses a driver which creates IO arrival requests for each disk device address in the configuration. Upon successful completion of the IO request, the driver creates another IO request in the same clock unit. Such a driver will tend to saturate either the disk devices or the data paths, depending on the IO data transfer sizes and the actual subsystem configuration. We capture the maximum number of start subchannels (SIO) per second as well as subsystem effective data rate as seen on the channels.

MAXIMUM THROUGHPUTS WITH FASTER CHANNELS: THE ADVENTURE CONTINUES

In this study we configure 10 gigabyte disk configurations of 1X and 2X drives (Figure 1) and vary data transfer speeds from 3 Megabytes per second (MB/s) to 4.5, 6, and 10 MB/s. Such transfer speeds are

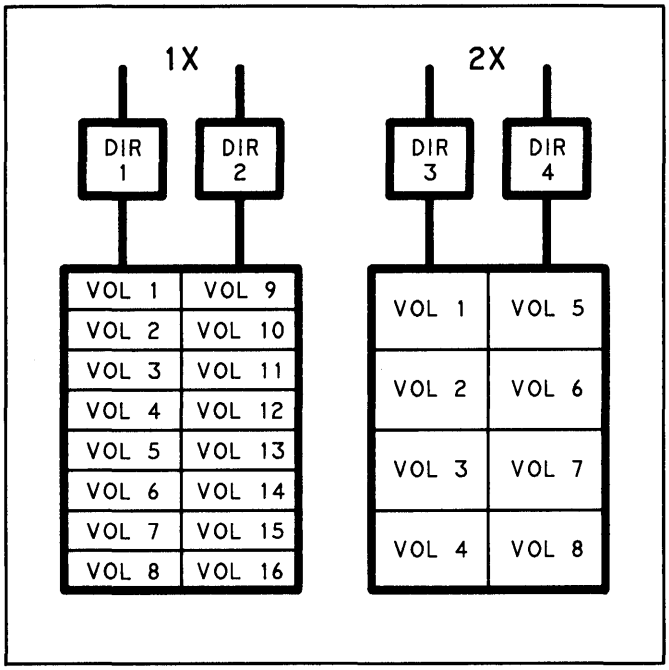


Figure 1. 10 GB DASD Configurations

considered to be valid design points in future disk products, likely with different form factors and not necessarily restricted to magnetic storage devices. Table 1 summarizes maximum throughputs across configurations using a pair of channels and a pair of directors. Note that the 1X disk configurations consist of four DASD storage boxes with a total of 16 actuators, and the 2X disk configurations consist of two DASD boxes with a total of 8 actuators.

Table 1. Maximum I/O Thruputs

2 x 16 1X Class

Transfer Speeds	4 KB Blocks At 50% Zero Seeks MB/s	Chans Busy	47 KB Blocks At 50% Zero Seeks MB/s	Chans Busy
3 MB/s	1.82	60.1%	5.19	94.1%
4.5	2.12	58.5%	6.65	83.4%
6	2.29	56.8%	8.23	80.3%
10	2.47	52.7%	11.58	74.5%

2 x 8 2X Class

Transfer Speeds	4 KB Blocks At 50% Zero Seeks MB/s	Chans Busy	47 KB Blocks At 50% Zero Seeks MB/s	Chans Busy
3 MB/s	1.37	45.5%	4.93	89.1%
4.5	1.44	39.8%	5.86	73.4%
6	1.49	36.9%	7.24	70.6%
10	1.56	33.3%	10.04	64.6%

Assumptions:
- Overhead I/O remains constant
- Closed loop mode
- Only data transfer portion allowed to vary

Data transfer sizes for all IOs are initially 4 KB and then 47 KB, the former representing nominal online and interactive IO and the latter representing large and possibly chained batch or perhaps disk to tape backup IO. The percentage of zero seeks in each simulation is statistically held at 50%. Sequentiality in disk IO patterns is a great influencer of zero seeks occurring with requests and results in increased data throughput. Looking at the cases in Table 1 we observe:

a) The 1X configurations always deliver more subsystem MB/s than do the 2X 10 Gigabyte configurations. This is because 16 read/write servers with 1X can do more work than 8 read/write servers with 2X configurations. In general the 1X configurations deliver 33% to 58% more data than the 2X boxes, as channel transfer speed is raised from 3 and up to 10 MB/s.

b) With 4 KB data transfer sizes, the overall improvements in throughputs are lower than what would be intuitively expected. This is because 4KB is a relatively small portion of a full disk track capacity, particularly with the faster assumed disk device transfer speeds. For instance a hypothetical DASD device capable of transmitting at 6 MB/s implies double linear track capacity (ie, 2 x 47.4 KBs, or 94.8 KB).

c) Protocol overhead accounts for over 50% of the channel utilization at conventional channel speeds of 3 MB/s; it takes 1.36 msec to transfer 4KB and an additional 1.4 msec for protocol overhead. More than one-half of channel and director connect time is consumed for non-overlapping protocol overhead.

d) Faster channels, then, reduce only the data transfer portion of channel busy conditions. Small data transfer sizes prevent significant improvements in IO response times.

e) With online and database applications representing the major DP growth areas, physical IO transfer sizes will likely remain small and therefore realize minimal IO performance improvements with faster channels.

f) Traditionally DASD performance analysts look to reducing start IO queing, reducing arm seek motion delays, or reducing data path contention and rotational position sense (RPS) reconnect times. The 6 and 10 MB/s channels result in modest IO throughput improvements, more so with the 2X 10 GB configurations.

Dramatic throughput improvements are realized when data transfer sizes are 47 KB. The 1X configurations continue to yield more data volume than their 2X counterparts, but the difference is trimmed to about 5%. Channel busy percentiles also rise with the 47 KB transfer sizes.

Sustained throughput increases with the faster device transfer speeds. Unlike the 4 KB scenarios, 47 KB transfer times constitute a larger proportion of total IO service time and stand to gain significant time reductions with 4.5, 6 or 10 MB/s transmission speeds. If the 1X and 2X configurations run with 10 MB/s channels, sustained throughput more than doubles when compared with conventional 3 MB/s channels. Figure 2 summarizes the maximum through-puts discussed above.

Impressive as the large transfer size throughputs are, the impact on overall system improvement, as related to faster channels of the future, is surprisingly limited. The storage application growth for commercial data processing is precisely where the performance gain of faster channels is at its worst, namely the online database and interactive environments.

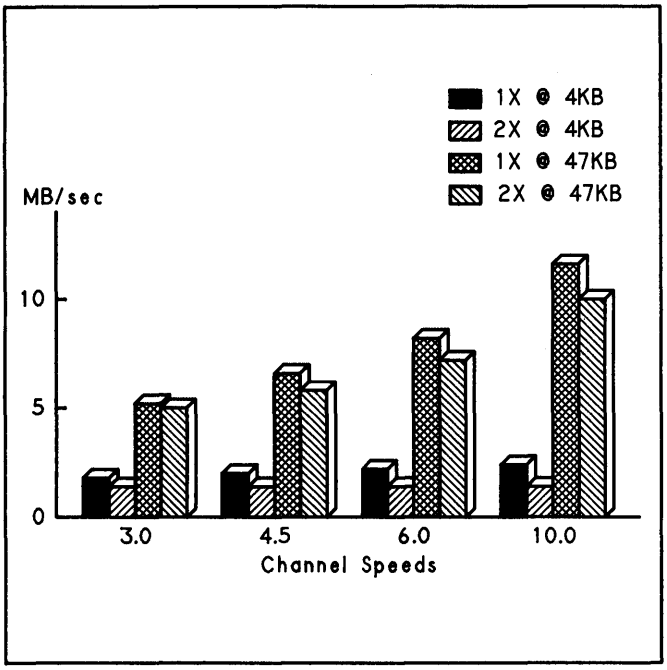


Figure 2. Maximum Throughputs
4 KB vs. 47 KB Transfer Sizes

Figure 3 depicts past and future storage requirements for batch, database, and system data [3]. Based on GUIDE customer surveys, databases account for 42% of disk volume usage in 1978, 55% in 1984, and 68% by 1990. Conversely, batch applications shrink from 40% in 1978, to 24% in 1984 and down to 8% by 1990. For the same time periods, total gigabytes of DASD for large installations increase from 9 GB, to 102 GB, and a sizable projected value of 1157 GB by 1990. The reality is that database

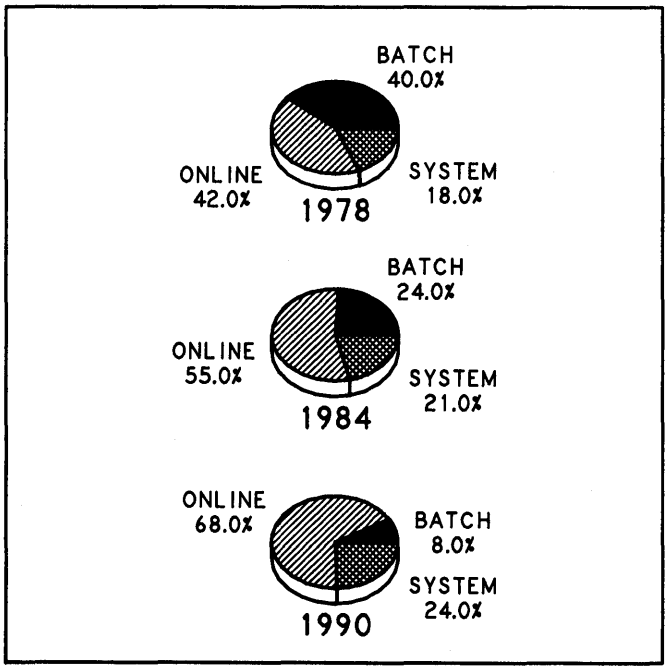


Figure 3. DASD Storage Growth
GUIDE Survey Data

and interactive users simply do not require more than 2 KB or 4 KB of data at a time. This is especially true when the data has a random pattern coupled with a low locality of reference profile. Relational databases, for example, use small IO data transfer sizes.

Throughout this analysis the IO workloads were stochastically balanced across all available volumes. Skewed IO workloads have a significant impact on throughput and response times, the details of which have been presented in a previous paper [1].

We conclude that faster data transmitting devices are well suited for large IO blocksize applications, typically being batch in nature or non-productive utility activity such as copying data from disk to tape or another disk. We further conclude that an increase in the number of channels supported will accommodate the need for more parallelism in the commercial application mainframe arena. Singh, King, and Anderson discuss the increasing importance of overlapping functions and maintaining a balance between processors and storage subsystems [4].

QUAD PORT ANALYSIS

We consider 10 GB disk configurations with 1X and 2X devices, using a total of four channels, four directors and four heads of string. The base cases represent a pair of dual ported configurations and are contrasted to quad port subsystems which allow any of the four data paths to access any disk device address.

The open loop model uses a driver generating an aggregate average of 150 SIO/sec. Response time results appear in Table 2 and Figure 4. We vary channel speeds and data transfer sizes. Stochastically random seek arm motion occurs 50% of the time, with zero seek times for the remaining 50%.

Table 2. 150 SSCH/sec Qual Port Analysis
Average Response Times

1X 4 x 16 Configuration

TRANS SPEEDS:	3 MB/s		4.5 MB/s		6 MB/s		10 MB/s	
BLOCK SIZE	w/o Qd ms	w/Qd ms	w/o Qd ms	w/Qd ms	w/o Qd ms	w/Qd ms	w/o Qd ms	w/Qd ms
2 KB	20.2	20.0	20.0	19.8	19.8	19.7	19.6	19.5
4 KB	21.2	20.9	20.6	20.3	20.2	20.0	19.9	19.7
8 KB	23.4	22.5	21.9	21.4	21.2	20.9	20.4	20.2
47 KB	44.7	39.1	24.0	31.3	29.6	27.8	24.8	23.9

2X 4 x 8 Configuration

TRANS SPEEDS:	30 MB/s		4.5 MB/s		6 MB/s		10 MB/s	
BLOCK SIZE	w/o Qd ms	w/Qd ms	w/o Qd ms	w/Qd ms	w/o Qd ms	w/Qd ms	w/o Qd ms	w/Qd ms
2 KB	25.7	25.6	25.2	25.1	24.9	24.9	24.6	24.5
4 KB	26.8	26.4	26.0	25.9	25.8	25.6	25.1	25.0
8 KB	30.0	29.3	27.7	27.2	26.8	26.4	25.8	25.7
47 KB	56.6	50.2	40.7	38.5	35.2	33.7	29.3	28.6

Qd = Quad Port

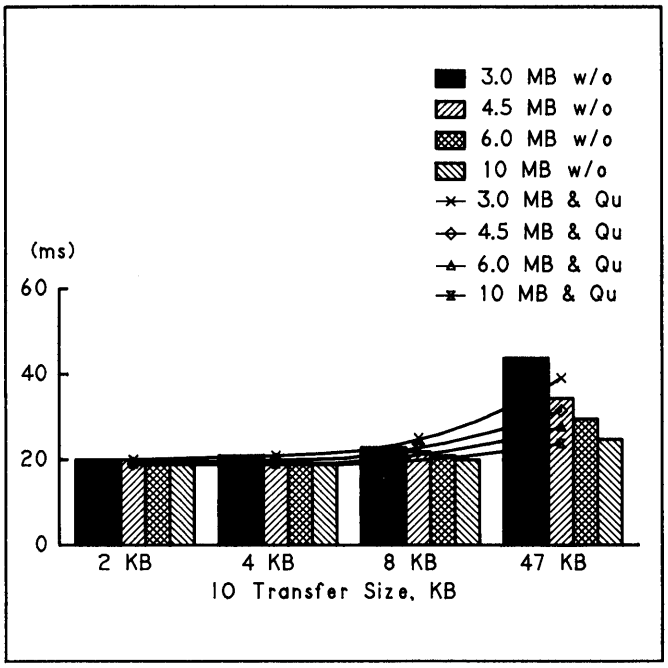


Figure 4. Quad Port Impact on Response Time
1X, 4X16, 150 SIO/sec

A number of interesting observations are made:

a) Quad port is more effective with 3 MB/s channels than with the faster channels. The faster channels, by definition, will have lower utilization, and therefore less probability of being unavailable. This suggests alternate data paths may not be used as frequently with the 4.5, 6, and 10 MB/s channels as they are for conventional 3 MB/s datastreaming channels.

b) The 2KB data transfer size IO response times realize no performance improvement with quad ports. Small physical block transfer sizes tend to make the disk drives the bottlenecks, not the channel and director data paths. In the 4KB cases, as well, hardly any improvement in response time occurs. Only in the 47 KB cases are there significant response time improvements.

c) The 47 KB blocksizes represent a full track of today's 1X or 2X devices. At 3 MB/s channel speeds, response time improves by 12.5% for the 1X configuration. With 10 MB/s channels, the quad port improvement reduces to about 3%.

d) We conclude that quad ports offer little performance improvement for blocksizes of 8 KB or smaller, and that faster future channels gain very little from quad ports.

While the performance associated with quad ports may not be dramatic on rotating disks, solid state disk subsystems stand to gain considerably, especially when channel utilizations are consistently high (in the 60% to 80% range). With the exception of nominal channel interface protocol and search overhead time, literally all solid state disk device service time is connected data transfer time. Faster channels will have significant improvements for paging and non-paging data sets residing on solid state IO devices. Figure 5 depicts the impact of faster channels on IO response times for a 1X cached subsystem of rotating disks versus an identical workload on a solid state disk subsystem. The cached configuration and workload characteristics are described in detail under the 'Cached Controller and Faster Channels' section of this paper. Note the significant percentage of improvement in IO response time as channel data transfer speed increases. Conversely the rotating disks with faster cache controller transfer speeds realize marginal response time improvements.

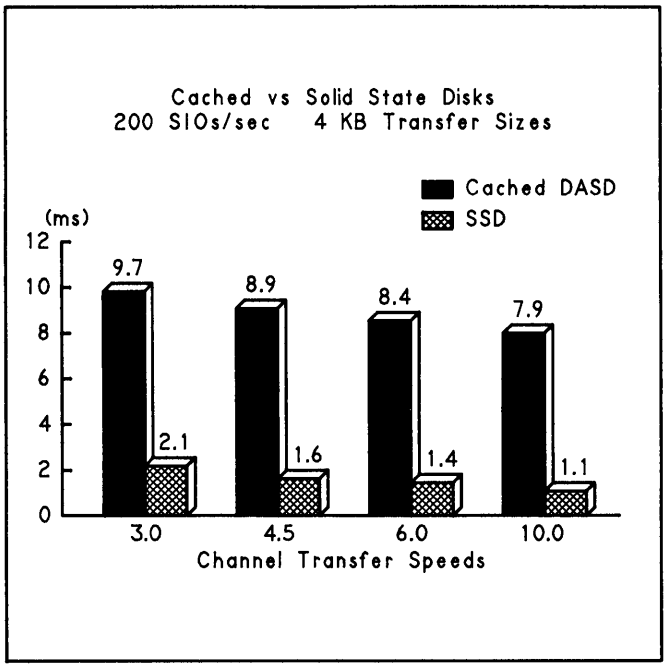


Figure 5. Faster Channels & I/O Response

Performance for rotating disks will not be the main attraction with quad porting; data connectivity will likely be the greatest advantage of quad ports. Instead of supporting one alternate data path to a given disk volume, future versions of operating systems will enjoy three alternate paths.

ONLINE AND BATCH IO RESPONSE TIMES

Table 3 summarizes average IO response times for two-director subsystems while varying data transfer speeds. IO workload rate is held constant at 150 SIO/sec with a Poisson distribution modified arrival rate. The 2 KB and 4 KB physical IO transfer sizes represent nominal blocksizes for online and interactive data access environments, while the 8 KB transfer sizes represent batch IO requests.

Small online data transfer sizes result in minimum improvements in response times, for reasons discussed previously. Channel utilizations show more significant improvements as the same IO workload is run on progressively faster channels and devices. The 1X configurations continue to outperform their 2X counterparts because sixteen actuators continue to be better than eight when the drives are the subsystem bottlenecks.

With 8 KB transfers, the faster transfer speeds show more improvement in IO response times. Larger blocksizes suggest that the data transfer ('connect time' in RMF terminology) component accounts for a significant portion of total IO service time. Faster channels and transfer speeds account for a drop in response time from 27.2 ms with 3 MB/s to 21.5 ms with 10 MB/s -- a 21% improvement. The equivalent 2X 10 Gigabyte configurations have worse response times but show comparable percentage gains with faster data paths.

Table 3. 150 SSCH/sec / 2K 4K 8K / 3, 4.5, 6, 10 MB/sec
Average Response Times

1X 2 x 16 Configuration

DATA TRANSFER SPEED	2 KB		4 KB		8 KB	
	TR ms	CHANS	TR ms	CHANS	TR ms	CHANS
3 MB/s	21.7	15.2%	22.9	20.3%	27.2	30.6%
4.5	20.7	13.5	21.7	17.0	40.2	23.8
6	20.5	12.7	21.2	15.2	22.9	20.3
10	20.1	11.6	20.6	13.1	21.5	16.2

2X 2 x 8 Configuration

DATA TRANSFER SPEED	2 KB		4 KB		8 KB	
	TR ms	CHANS	TR ms	CHANS	TR ms	CHANS
3 MB/s	26.7	15.2%	28.9	20.3%	35.3	30.6%
4.5	26.0	13.5	27.5	17.0	30.6	23.8
6	25.8	12.7	26.7	15.2	28.9	20.3
10	25.4	11.6	26.1	13.1	27.2	16.2

TR = average response times
50% Zero Seeks

We conclude that batch applications with larger transfer sizes will see moderate response time gains with faster channels; batch IO workloads will likely experience more improvement than online or interactive applications. The biggest gain occurs where it is less needed, in terms of storage growth directions.

MORE VS FASTER CHANNELS

We compare IO response times at 150 SIO/sec for four channel subsystem configurations in Table 2 to response times for two channel configurations in Table 3. Looking at 8 KB transfers, the 1X configuration in Table 2 shows average response time of 23.4 ms with four conventional 3 MB/s channels. Table 3 shows 22.9 ms for two 6 MB/s channels, an insignificant difference of 0.5 ms. Note that in each configuration total channel bandwidth is constant at 12 MB/s. Similarly the 2 KB scenarios show even less response time improvement with 6 MB/s channels.

Aggregate IO subsystem performance is just as easily accomplished with increasing the number of conventional datastreaming channels as opposed to increasing device and data path transmission speeds.

CACHED CONTROLLERS AND FASTER CHANNELS

An architectural alternative is to support faster channel transfer rates only for cache hit data, and not for data read or written directly to rotating disk devices. In this analysis we consider four cache controller workload profiles:

1) IOs with a 9:1 ratio of reads to writes, and 90% read hit probability. This is designated as a "90/90" column header in Table 4.
2) IOs with a 9:1 read to write ratio and a 75% read hit probability, or "90/75" in Table 4.
3) IOs with a 4:1 read to write ratio and a 75% read hit rate, or "80/75".
4) A marginal cache controller potential with an IO pattern of roughly 2:1 reads to writes and 75% read hit rate, or "70/75".

All direct or native IO to the 1X and 2X disk configurations use the conventional 3 MB/s data transfer rate. 50% of all seeks are random with the remaining seek times being zero. Workload arrival rates to the subsystems vary from a light load of 50 SIOs per second, and increase in increments of 50, up to a maximum of 200 per second. Tables 4 and 5 summarize response times for 4 KB data transfer sizes, for 1X and 2X devices, respectively. Tables 6 and 7 summarize analagous response times for 2 KB transfer sizes.

Table 4. 4 KB: Average Response Time, Cached Controllers

1X 2 x 16 Configuration

TRANSFER SPEED	SSCH/sec	90/90	90/75	80/75	70/75
3 MB/s	50	6.0 ms	9.1 ms	10.2 ms	11.4 ms
3	100	6.9	11.2	12.5	13.4
3	150	7.8	16.5	16.9	18.2
3	200	9.7	30.5	29.3	28.4
4.5	50	5.6	8.7	9.9	11.2
4.5	100	6.4	10.8	12.0	13.3
4.5	150	7.2	15.3	16.1	17.2
4.5	200	8.9	26.9	25.4	26.9
6	50	5.4	8.5	9.7	11.0
6	100	6.2	10.6	11.8	12.9
6	150	7.0	15.1	16.1	17.0
6	200	8.4	26.0	24.4	25.9
10	50	5.2	8.3	9.6	10.8
10	100	5.9	10.3	11.5	12.6
10	150	6.6	14.1	15.5	16.4
10	200	7.9	22.6	23.4	25.2

Table 5. 4 KB: Average Response Time, Cached Controllers

2X 2 x 8 Configuration

TRANSFER SPEED	SSCH/sec	90/90	90/75	80/75	70/75
3 MB/s	50	6.5 ms	10.2 ms	11.3 ms	12.6 ms
3	100	7.8	13.2	14.8	16.1
3	150	9.3	20.7	22.3	23.3
3	200	12.3	42.9	40.8	44.7
4.5	50	6.1	9.7	11.0	12.3
4.5	100	7.5	12.6	14.3	15.6
4.5	150	8.8	19.6	20.9	22.2
4.5	200	11.0	38.7	38.8	39.9
6	50	5.9	9.7	10.8	12.2
6	100	7.1	12.5	14.1	15.5
6	150	8.4	18.6	20.4	22.2
6	200	10.6	36.6	27.5	38.3
10	50	5.7	9.5	10.6	12.0
10	100	6.8	12.1	13.9	15.4
10	150	8.1	17.8	19.7	21.1
10	200	10.3	32.1	35.2	35.1

We notice the longest average response times occur when the cache profiles are poorest -- the 70/75 scenarios described above. For example, in Table 4 for 3 MB/s transfer speeds response time at 200 SIO/sec is 9.7 ms averaged for all IOs with the 90/90 cache profile, compared to 28.4 ms with the 70/75 profile.

Table 6. 2 KB: Average Response Time, Cached Controllers

1X 2 x 16 Configuration

TRANSFER SPEED	SSCH/sec	90/90	90/75	80/75	70/75
3 MB/s	50	5.3 ms	8.3 ms	9.4 ms	10.6 ms
3	100	6.0	10.2	11.3	12.4
3	150	6.8	13.9	15.1	15.6
3	200	8.1	22.6	22.5	22.6
4.5	50	5.1	8.1	9.2	10.5
4.5	100	5.8	10.0	11.1	12.0
4.5	150	6.4	13.4	14.5	15.5
4.5	200	7.8	20.6	21.1	22.4
6	50	5.0	8.1	9.2	10.4
6	100	5.7	9.9	10.9	12.1
6	150	6.4	13.1	14.2	15.3
6	200	7.6	20.3	21.1	21.2
10	50	4.9	8.0	9.1	10.3
10	100	5.5	9.8	10.8	11.9
10	150	6.2	12.9	13.9	15.5
10	200	7.4	20.3	20.1	21.2

Table 7. 2 KB: Average Response Time, Cached Controllers

2X 2 x 8 Configuration

TRANSFER SPEED	SSCH/sec	90/90	90/75	80/75	70/75
3 MB/s	50	5.8 ms	9.4 ms	10.4 ms	11.7 ms
3	100	6.9	12.4	13.5	14.8
3	150	8.1	18.2	19.0	20.5
3	200	10.2	32.1	31.3	32.8
4.5	50	5.6	9.2	10.3	11.6
4.5	100	6.7	11.6	13.3	14.7
4.5	150	7.9	17.0	18.1	20.2
4.5	200	9.7	29.4	29.7	31.7
6	50	5.5	9.2	10.2	11.5
6	100	6.5	11.4	13.0	14.5
6	150	7.7	17.2	18.2	19.6
6	200	9.5	29.2	28.8	31.1
10	50	5.4	9.1	10.1	11.4
10	100	6.4	11.5	12.8	14.1
10	150	7.5	16.4	18.3	19.2
10	200	9.3	27.6	29.1	30.1

Of more interest is the overall small improvement with faster channels. For the same workload at 200 IOs per second, and our best cache case of 90/90, response time improved from 9.7 ms to 8.9 ms with 4.5 MB/s channel speed. 6 MB/s channels reduced response time an extra 0.5 ms to 8.4 ms, and 10 MB/s channels down to 7.9 ms.

As the cache profiles degrade, the performance impact of faster channels is further diminished with cache controller subsystems (Figure 6). In the marginal 70/75 cache scenarios, response time improvement at 200 IOs per second arrival workload is 11%, at best, and this is with 10 MB/s vs 3 MB/s channels. Channel interface protocol overhead remains constant at 1.4 ms, thus continuing to diminish improvements intuitively expected with faster channels.

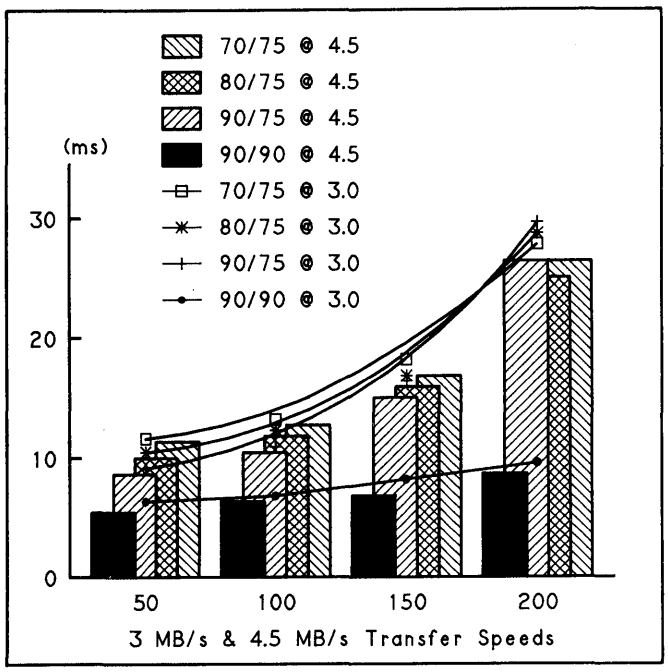


Figure 6. 1X DASD Response Times
4 KB, Cached, 2X16, 50% Zero Seeks

Table 5 summarizes average response times for the 2X 10 GB configurations. The double capacity devices consistently show longer response times for identical workloads and cache profiles, as compared to the 1X configurations of Table 4. Note how much more rapidly the decay in average IO response times occurs as the cache profile deteriorates from 90/90 to 90/75 and beyond. Reducing the number of servers by half with 2X disk devices, and incurring peaks of 200 IOs per second can have a debilitating affect on cached subsystem response time. Faster channels do not address the bottleneck, and the bottleneck is the device itself when data transfer size is 4 KB or smaller.

Using 2 KB data transfer sizes is of interest with cached IO subsystems. The notion of achieving more read hits per staged track with small physical blocksizes is appealing in a cached IO environment. For instance, in Table 6 the 200 SIO per second workload at 3 MB/sec results in an average response time of 8.1 ms. Compared to the same scenario in Table 4 with 4 KB blocksizes, response time improves by more than 16%.

Most of this improvement is attributable to the cache controller and the 90/90 IO workload profile. Increasing channel speeds with 2 KB transfer sizes results in marginal response time improvements, for reasons discussed previously. The 90/90 profiles can be attained by carefully selecting the right files for cached subsystems. Best file candidates are [3]:

* TSO: Libraries, ISPF, catalogs, user data
* IMS: ACB, PSB, MFS, ADF work data base, program libraries
* CICS: Program libraries
* General System: Partitioned data sets, PROCLIB, master and user catalogs, control data sets (RACF, DFHSM).

A point of interest on the cached subsystem response times. On Table 5, for example, note that response time for the 3 MB/s channel speeds and 200 IOs per second workload were actually better for the 80/75 cache profile than for the 90/75 cache profile. At first, we might

expect the profile with the better read to write ratio of IOs to have better response times, the opposite of what occurred. The 90/75 profile has a response time of 42.9 ms, as opposed to the 40.8 ms of the 80/75 profile.

This phenomenon is explainable when we consider the expected number of disk to cache track stages. Multiplying 200 SIOs per second by .9 we determine the number of read IOs to be 180; similarly the 80/75 profile results in only 160 read IOs. Since we have cache hit rates of 75% in each case, we can expect to incur misses the remaining 25% of the time. Misses, by design, result in cache track stages; there will actually be more cache track staging in the 90/75 case than in the 80/75 case: 45 stages vs 40 stages, respectively (Table 8). In heavily loaded subsystems, the effect on overall performance can indeed be worse with 'better' cache workload profiles. The weakness of cache controllers is the increased device to cache director utilizations. The impact of a cached subsystem with insufficient megabytes of memory is more important at times than the ratio of reads to writes.

Table 8. Cache Controller Results, Tables and Graphs

Symbol	Definitions
	90/90 9% Reads, 10% Writes, and 90% probability of cache read hits
90/75	90% Reads, 10% Writes, and 75% probability of cache read hits
80/75	80% Reads, 20% Writes, and 75% probability of cache read hits
70/75	70% Reads, 30% Writes, and 75% probability of cache read hits

Example

Case A: At 20 SSCH/sec, and 80/75 profile:
Estimated # Reads = 200 x (.8) = 160 reads/sec
Estimated # Writes = 200 x (.2) = 40 writes/sec
Estimated # Read Stages = 160 (1.0 - .75) = 40 Cache States/sec

Case B: At 200 SSCH/sec, and 70/75 profile:
Estimated # Reads = 200 x (.7) = 140 reads/sec
Estimated # Writes = 200 x (.3) = 60 writes/sec
Estimated # Read Stages = 140 (1.0 - .75) = 35 Cache States/sec

Observation: Case A, with a better read to write ratio than Case B, will experience more data path contention than Case B.

OTHER MODEL ASSUMPTIONS

- 50% zero seek arm movements
- Poisson arrival rate distributions
- GPSS V Monte Carlo Discrete Event Simulation
- Data transfer sizes held constant within a simulation run

We can state that traditional cache workload characteristics influence subsystem performance considerably more than faster channels will. Choosing the best file candidates for cached devices continues to be a key factor in improving performance.

CONCLUSION

Faster data transfer times on rotating disk subsystems will probably have maximum benefit in the slowest growing portion of the data storage environment -- batch applications with large or chained IO blocks. To achieve significant throughput and response time improvement, large data transfer sizes must be exploited.

Solid state IO subsystems stand to gain more from faster channels, even with small data transfer sizes, since the bulk of their total IO service time is electronic data transfer, without any mechanical and disconnected overhead times. Quad porting has maximum performance gain with solid state systems, but the dominant feature with quad porting is connectivity.

Single and double capacity disk devices perform well, however there are performance differences when IO workload intensity and data transfer sizes are varied. A mix of both device types with and without cache controllers is desirable, allowing an informal and transparent hierarchy of isolated disk pools to accommodate the needs of batch, interactive, and online file applications.

The adventure of improving IO performance is far from over! Configuration tuning via hardware selection and file placement continues to offer plenty of opportunity in providing better IO service times as newer and faster processors are introduced.

REFERENCES

[1] B. Duhl: Notes On Performance Considerations For Single and Dual Density DASD; CMG 1986 Proceedings, pp 710 - 715.

[2] GPSS V Users Manual; IBM # SH20-0851.

[3] MVS Extended Architecture Storage Management Library; IBM # GC26-4262.

[4] Y. Singh, G. M. King, J.W. Anderson: IBM 3090 Performance: A Balanced System Approach; IBM Systems Journal Vol. 25, No. 1, 1986.

An Intuitive Approach to Performance Prediction with Application to Workload Management in VM SP/HPO

Joseph Hellerstein

IBM Thomas J. Watson Research Center

ABSTRACT

Workload management can be greatly facilitated by using predictive models to anticipate the impact on system resources when workloads are assigned to computer systems. In the IBM operating system VM SP/HPO (Virtual Machine System Product with High Performance Option), contention for the first 16 megabytes of storage is often a major performance problem, and its resolution often requires workload management. Such storage contention is indicated by three variables: page steals from the low demand paging area, (LOSTEALRAT), free storage extends (FREXTND), and prime free utilization (PRIMEUTL). We know of no existing predictive model for these variables. Herein, we outline our efforts to use a statistical approach to construct such a model, and we present an apparently new technique which has been developed in the process - **non-parametric interpolative-forecasting for monotone functions (NIMF)**. Although NIMF is discussed in the context of VM SP/HPO performance analysis, it is a generally applicable statistical technique. Indeed, we have used NIMF to analyze data from MVS/XA systems.

Our attempts to use least squares regression to predict LOSTEALRAT have met with little success, primarily because of an inability to find an algebraic expression relating LOSTEALRAT to the workload variables (number of logged-on users and total virtual machine IO rate). However, the regression studies do indicate that LOSTEALRAT is monotonically increasing in the workload variables. When such a monotone relationship exists, an intuitive approach often used by performance analysts is to bound a predicted value by using the monotone relationship to select a lower and upper bound from the measurement data. This idea motivated the development of NIMF. A NIMF model only requires specifying a monotone relationship between the independent and dependent variables. Using data from several VM SP/HPO 4.2 systems, NIMF and regression are compared as to their accuracy of predicting contention for the first 16 megabytes of storage. These studies indicate that NIMF can be as accurate as a good regression model. However, formulating NIMF models is much easier than formulating regression models since NIMF only requires specifying a monotone relationship between independent and dependent variables while regression requires an algebraic relationship. We conclude by using NIMF to determine how workload can be managed to control LOSTEALRAT in a large VM SP/HPO system.

1. Introduction

An important aspect of managing a computer installation is workload management which involves assigning workloads (e.g., interactive users) to computer systems so that system resources (e.g., CPU, IO devices, storage) are not over-utilized. Since moving workloads is time consuming and disruptive, it is desirable to have tools which can predict the resource impact of increasing or decreasing workload. Numerous pre-packaged models exist for predicting the utilizations of CPUs, channels, control units, and devices (e.g., BEST/1 [BGS 84], VM Performance Planning Facility [IBM 84]). VM SP/HPO (Virtual Machine System Product with High Performance Option) is an IBM operating system which is widely used for interactive computing. Large VM SP/HPO systems are often constrained by contention for the first 16 megabytes of storage. Unfortunately, existing computer-performance prediction models are not adept at predicting storage contention. Indeed, we know of no existing model which predicts the effects of contention for the first 16 megabytes of storage in VM SP/HPO. Herein, we describe a statistical approach to predicting such storage contention. This approach is based on an apparently new technique, **non-parametric interpolative-forecasting for monotone functions (NIMF)**, which is described in this paper. Although NIMF is presented in the context of analyzing VM SP/HPO performance, NIMF is a generally applicable statistical technique. Indeed, we have used it to analyze data from MVS/XA systems.

A major reason for the absence of an existing model for contention for the first 16 megabytes of storage is that the commonly used techniques for constructing predictive models (e.g., queueing theory [Kleinrock 75], operational analysis [Denning and Buzen 78], and regression [Draper and Smith 68]) require formulating an algebraic relationship between the dependent variables (e.g., contention for the first 16 megabytes of storage) and the independent variables (e.g., workload). Unfortunately, contention for the first 16 megabytes of storage has a complex relationship to workload, and this is difficult to formulate as an algebraic expression. However, we do expect storage contention to increase with workload. This observation has motivated the development of NIMF. A NIMF model only requires specifying a **monotone relationship** which indicates the directional effect on the independent variable (i.e., increases or decreases) when a dependent variable increases. NIMF models are easy to formulate, and our experience suggests that their accuracy is comparable to that of a good regression model.

The remainder of this paper is organized as follows. Section 2 discusses contention for the first 16 megabytes of storage in HPO 4.2. Section 3 outlines our attempts to construct regression models for one aspect of contention for the first 16 megabytes of storage (low-DPA page steal rates). These attempts were not successful primarily because we could not find a good algebraic expression relating low-DPA page steal rates to the workload variables. NIMF is described in section 4. Section 5 presents several case studies in which the accuracy of NIMF is compared to that of regression for predicting contention for the first 16 megabytes of storage. Section 6 gives an overall assessment of NIMF. Section 7 applies NIMF to managing workload in a large VM SP/HPO system so as to control low-DPA page steal rates. Our conclusions are contained in section 8. Appendix A summarizes the symbols used in this paper.

This paper is targeted for two audiences - computer performance practitioners and data analysis experts. As such, sections are written at different levels of detail. Specifically, computer performance practitioners may prefer skipping the mathematical details in the latter part of section 4.1; the examples provided in 4.1 should be sufficient to illustrate the NIMF technique. On the other hand, data analysis experts may be particularly interested in the detailed discussion contained in section 4.1.

2. Low-Memory Storage Contention in VM SP/HPO 4.2

High-end mainframes have large amounts of main memory (e.g., 64 megabytes), and there may be expanded storage as well (see Fig. 1). However in VM SP/HPO, the operating system (CP) uses 24 bit addressing and so can directly address only the first 16 megabytes of main memory. Thus, CP control structures must have addresses less than 16 megabytes regardless of the size of main memory. In high-end systems, this restriction can create contention for the first 16 megabytes of storage. Three key components of this storage in VM SP/HPO 4.2 are:

- Low Demand Paging Area (DPA) - used by a variety of system services (e.g., Inter-User Communications Vehicle)
- Free Storage Area - used by CP for control structures such as swap tables, virtual device tables, and pointers to spool files
- Prime Free Storage - a sub-area of free storage which is used by CP as a fast-path through the free-storage buffer manager

Low DPA, free storage, and prime free storage have dynamic demands placed on them; so from time to time their configured sizes are insufficient to satisfy these demands. An indication that low DPA is too small is when a page request must be satisfied by stealing a page from a user in the multi-programming set (referred to as an in-queue user); the rate at which this occurs is the **low-DPA page steal rate**. When free storage is too small, a page is allocated from low DPA. This is called a free storage extend, and the rate at which this occurs is the **free storage extend rate**. Extends are undesirable because they: (a) cause additional CP overhead and (b) increase spin-lock waits in multi-processor systems. When prime free storage is too small, a prime miss occurs and the buffer request must be satisfied by non-prime free storage. Doing so increases CP overhead. We assess contention for prime free storage by **prime free utilization** which is the fraction of prime free storage which is allocated. Thus, contention for the first 16 megabytes of storage can be gauged by looking at the low-DPA page steal rate (denoted by the variable **LOSTEALRAT**), free storage extend rate (denoted by the variable

Fig. 1: Overview of VM/HPO 4.2 Systems Studied

Characteristic	VMSYS1	VMSYS2	VMSYS3
CPU	3090-200	3081K	3081K
Main Memory	64 Meg.	64 Meg.	64 Meg.
Expanded Store	128 Meg.		
Workload	CMS	CMS	PROFS

CMS = Conversational Monitor System
PROFS = Professional Office System

FREXTND), and prime free utilization (denoted by the variable **PRIMEUTIL**).

Sometimes contention for the first 16 megabytes of storage can be significantly reduced by tuning. For example, if the low-DPA page steal rate is large (e.g., > 5/second), and there are no extends or prime free misses, then the free storage area can be reduced [Hoag 86]. However, when there are low-DPA page steals, extends, and prime misses, tuning becomes much more problematic and workload management should be considered.

But how much must the workload be reduced to achieve a specified level of contention? Since moving workloads is time consuming and disruptive, there is benefit in predicting what the level of contention for the first 16 megabytes of storage would be after reducing workload by a specified amount. Constructing such a predictive model requires first characterizing the workload. Here we use the following high-level characterization:

1. Fixed demands imposed by virtual machines (e.g., swap tables and virtual devices).
2. Variable demands due to the activity level of the virtual machines (e.g., CP service requests and CCW translation).

Our measure of fixed demands is the number of logged-on virtual machines[1] (**LOGGED**), and our measure of variable demands is the total VIO rate of all virtual machines (**VIO**). The Virtual Machine Monitor Analysis Program (VMMAP) reports FREXTND, PRIMEUTIL[2], LOGGED, and VIO [IBM 85]. LOSTEALRAT is computed from two variables reported by VMMAP:

$$\text{LOSTEALRAT} = \text{STEALRATE} - \text{HISTEALRAT} \quad (1)$$

Where:
STEALRATE = Total page steal rate (above and below 16 megabytes)
HISTEALRAT = Page steal rate for pages above 16 megabytes

Fig. 1 characterizes three VM systems from which data were collected. For each system, data were collected on several days with variable values averaged over fifteen minute intervals. We use these data to evaluate models for predicting the dependent variables LOSTEALRAT, FREXTND, and PRIMEUTIL from the independent variables LOGGED and VIO.

3. Attempts to Find a Regression Model for Low-DPA Page Steal Rate

Least squares regression (hereafter, just regression) is a commonly used statistical technique in which an algebraic expression is fit to a set of measurement data. We represent measurement instances as ordered pairs of the form $(y_1, X_1), \ldots, (y_I, X_I)$ where y_i is the i-th instance of the **dependent variable** to predict (e.g., LOSTEALRAT_i) and $X_i = (x_{i1}, \ldots, x_{iJ})$ is a vector of **independent variables** or predicting variables (e.g., $(\text{LOGGED}_i, \text{VIO}_i)$). Regression requires that the dependent variable be related to the independent variables by an algebraic expression. The general form is

$$y_i = f(X_i) + \varepsilon_i \quad (2)$$

Where:
f = function with a known form
$\varepsilon_1, \ldots, \varepsilon_I$ = error terms which are independent random variables from the same normal distribution with mean 0

For example, if LOSTEALRAT is hypothesized to be a linear function of LOGGED and VIO, the regression equation is:

$$\text{LOSTEALRAT}_i = b_0 + b_1\text{LOGGED}_i + b_2\text{VIO}_i + \varepsilon_i$$

[1] Of course the number of logged-on users does not account for all fixed demands (e.g., spool files) nor does it account for effects due to virtual machine size (e.g., swap tables). However, our experience is that it is a good approximation.

[2] VMMAP does not assign a variable name to prime free utilization. PRIMEUTIL is a name is of our own creation.

Fitting a least squares regression model to a set of measurement data involves choosing the b_j so as to minimize (in a least squared sense) the deviation of y_i from its estimated value. When a regression model is fit to a set of measurement data, the quality of the fit can be assessed by R^2 which is the fraction of the variability in the data which is accounted for by the regression model. We want R^2 to be as close as possible to 1; a reasonable model should have R^2 much larger than .5.

Using regression to do predictions involves: (1) having a set of **base measurement data** from which each b_j is estimated and (2) computing an estimate for an unknown y^* by substituting the values for its associated X^* into the fitted regression equation. In addition, we want a range of y values that contains y^* with a specified level of confidence. This is referred to as a **prediction interval** and is specified by a lower bound, y_L, and an upper bound, y_H. The probability of y^* being within the prediction interval is the **confidence level** of the prediction interval. We write the confidence level as $1-\alpha$ (e.g., for a 75% confidence level, $\alpha = .25$). Thus, $Pr[y_L \le y^* \le y_H] \ge 1-\alpha$. We obtain prediction intervals for regression by constructing confidence intervals for the expected value of y^*. (See [Draper and Smith 68] for a discussion of regression confidence intervals.) To summarize:

Obtaining Prediction Intervals Using Regression

Given

$y_1, \ldots, y_I$ = dependent variables with $y_I = f(X_I) + \varepsilon_I$
$X_1, \ldots, X_I$ = vectors of independent variables with $X_I = (x_{I1}, \ldots, x_{IJ})$
f = function with a known form
$\varepsilon_1, \ldots, \varepsilon_I$ = error terms which are independent random variables from the same normal distribution with mean 0
$1-\alpha$ = confidence level
X^* = values of independent variables for an unknown y^*

Find

y_L and y_H such that $Pr[y_L \le y^* \le y_H] \ge 1-\alpha$.

Building a regression model for LOSTEALRAT begins with attempting to formulate an algebraic expression relating LOGGED and VIO to LOSTEALRAT. Unfortunately, we have little intuition about the form of such an algebraic expression. Fig. 2 shows LOSTEALRAT plotted separately against LOGGED and VIO. We see no obvious algebraic form for the relationship, although it does appear that LOSTEALRAT is increasing in both LOGGED and VIO.

Lacking intuition on a specific algebraic expression, we use a very general approach - approximating an unknown function by a polynomial [de Boor 78]. Our strategy is to consider polynomials of increasingly higher degree until there is little increase in R^2. We consider the following models:

R1: $y = b_0 + b_1L + b_2V + \varepsilon$
R2: $y = b_0 + b_1L + b_2V + b_3L^2 + b_4LV + b_5V^2 + \varepsilon$
R3: $y = b_0 + b_1L + b_2V + b_3L^2 + b_4LV + b_5V^2 + b_6L^3 + b_7L^2V + b_8LV^2 + b_9V^3 + \varepsilon$

Where:
y = dependent variable (LOSTEALRAT in this case)
L = LOGGED
V = VIO

Fig. 3 summarizes the results of evaluating R1, R2, and R3 for the VMSYS1 base measurement data using GRAFSTAT [Schatzoff and Lane 86]. After exploring a variety of data transformations as recommended in [Daniel and Wood 80] (e.g., taking logarithms, excluding outliers), the highest R^2 is still fairly small[3]-only 37%. Also, while there is a big increase in R^2 from R1 to R2 (25% to 35%), the increase from R2 to R3 is quite small (35% to 37%); this gives us little confidence of improving the model by using polynomials of higher degree. Furthermore, it is hard to interpret these models. For example in R3, why is V^2 significant while V is not, and why is LOSTEALRAT decreasing in $I.V^2$?

If there exists a good algebraic expression relating LOSTEALRAT to LOGGED and VIO, R^2 could be increased. Possibly if sufficient effort were expended such an algebraic expression could be formulated. However, Fig. 4 suggests that knowing this would not help with discovering an algebraic relationship be-

[3] Actually, some transformations do increase R^2 (e.g. $\sqrt{y}$), but the ones that we investigated also increase the width of the prediction intervals.

FIG. 2: LOSTEALRAT VS. INDEPENDENT VARIABLES FOR VMSYS1
(EACH POINT IS A 15 MINUTE INTERVAL)

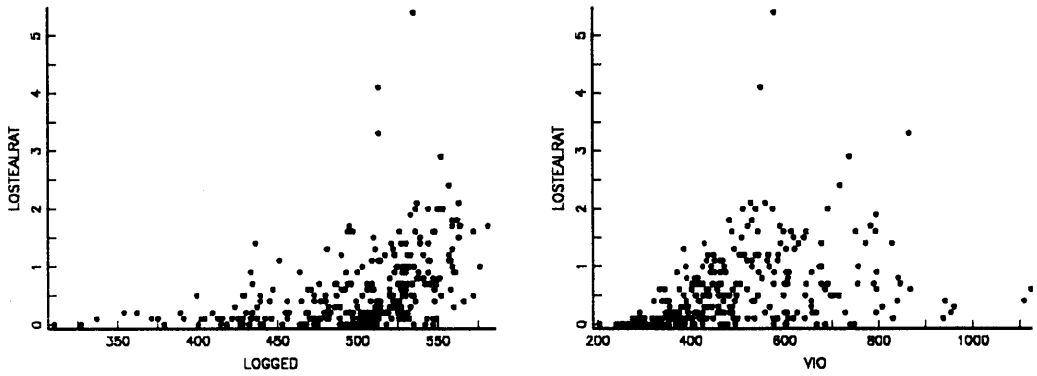


Fig. 3: Regression Studies for Low-DPA Page Steal Rate on VMSYS1

Term	Value of Coefficient in Each Model		
	R1	R2	R3
constant	**-2.62**	**6.25**	-.107
L	**.0053**	**-.031**	.1052
V	**.0011**	.000374	-.025
L^2	.	**.000032**	−.00028
LV	.	**.000012**	−.00028
V^2	.	**−.000043**	**.000019**
L^3	.	.	2.1×10^{-7}
L^2V	.	.	2.0×10^{-8}
LV^2	.	.	**-5.7×10^{-8}**
V^3	.	.	2.6×10^{-9}
R^2	.26	.34	.37

L = LOGGED (number of logged-on users)
V = VIO (total virtual machine VIO rate)
bold = significant at .1 level

tween FREXTND and LOGGED, VIO or an algebraic relationship between PRIMEUTIL and LOGGED, VIO. Thus, still more effort would be necessary to construct these models. Even more discouraging is the concern that the specific algebraic relationship may change with the software release and patches applied to the system. These observations make us very pessimistic about finding an acceptable regression model for predicting contention for the first 16 megabytes of storage based on LOGGED and VIO.

4. Non-parametric Interpolative-forecasting for Monotone Functions (NIMF)

Our difficulties with regression stem from its requiring us to provide something that we do not know - an algebraic expression relating LOSTEALRAT to LOGGED and VIO. However, we do believe that LOSTEALRAT (as well as the other measures of contention for the first 16 megabytes of storage) increases with LOGGED and VIO.

Ideally, we want a technique which produces prediction intervals but which only requires specifying directional effects (e.g., LOSTEALRAT increases with LOGGED and VIO). Informally, a **monotone relationship** specifies the direction in which the independent variables must change in order to increase the dependent variable. For example,

MR1: LOSTEALRAT increases when at least one of LOGGED and VIO increase and neither decreases

We say that y is a **monotone function** of X if there is a monotone relationship between y and X. More formally, a function f is monotone if for all j the derivative of f with respect to x_j, $\frac{\partial f}{\partial x_j}$, never changes sign.

Our central observation is that **monotone relationships can be used to bound a predicted value based on measurement data.** This is best illustrated by example.

Example 1: Obtaining Prediction Intervals using Monotone Relationships

We can bound an unknown value of LOSTEALRAT (denoted by LOSTEALRAT*) when LOGGED$^* = 500$ and VIO$^* = 500$ based on MR1 and the following data:

Entry #	LOGGED	VIO	LOSTEALRAT
1	482	486	.3
2	510	611	1.5

Specifically, if we assume that MR1 always holds, then a lower bound for LOSTEALRAT* is LOSTEALRAT$_1$ (= .3) since:

1. LOSTEALRAT is increasing in LOGGED and VIO (MR1)
2. LOGGED$_1$ = 482 < 500 = LOGGED* and VIO$_1$ = 486 < 500 = VIO*.

Similarly, LOSTEALRAT$_2$ (= 1.5) provides an upper bound since LOGGED$_2$ = 510 > 500 = LOGGED* and VIO$_2$ = 611 > 500 = VIO*.

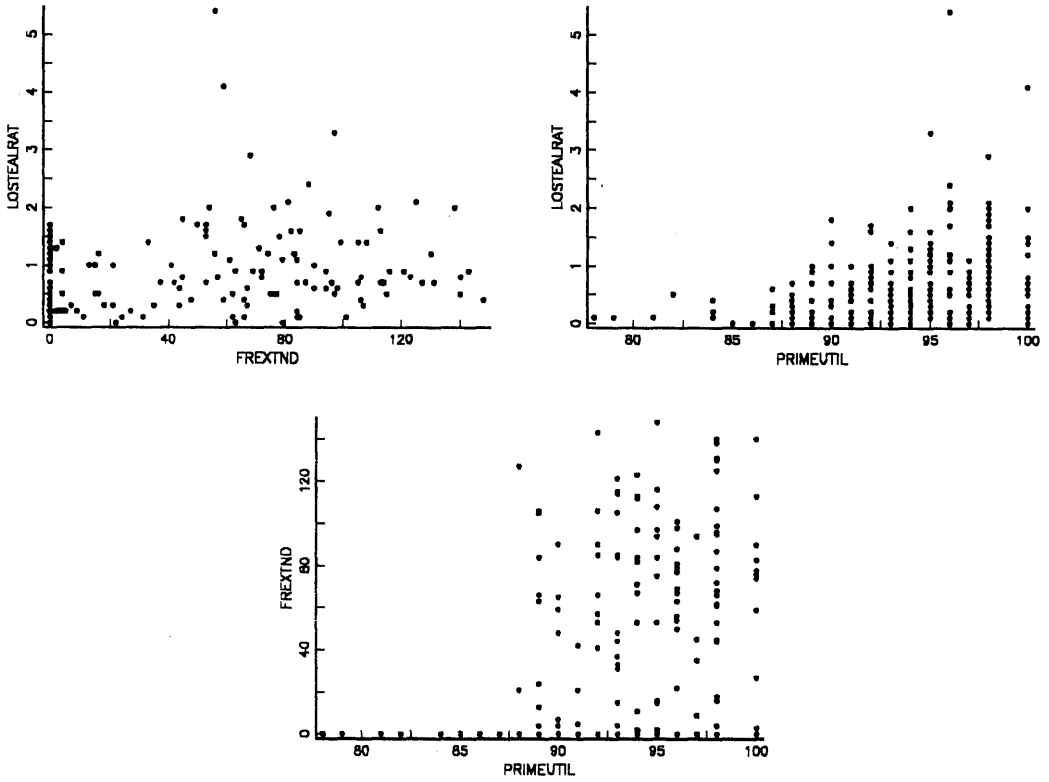
The idea behind example 1 is that there is an underlying function f which relates LOSTEALRAT to LOGGED and VIO, or LOSTEALRAT = f(LOGGED, VIO). Since we do not know the functional form of f (e.g., LOSTEALRAT = −2.62 + (.0053)LOGGED + (.0011)VIO), example 1 illustrates how LOSTEALRAT* can be bounded using only our intuition that f produces larger values when both LOGGED and VIO increase.

We formalize example 1 as follows. We assume that f is a monotone function. Let δ_j indicate the directional effect of x_j on $f(X)$, or

$$\delta_j = \text{sign of } \frac{\partial f}{\partial x_j}$$

Let $\Delta = (\delta_1, \ldots, \delta_J)$. We **denote a monotone relationship by** $y(\delta_1 x_1, \ldots, \delta_J x_J) = y(\Delta X)$. For example, MR1 is denoted by LOSTEALRAT(+LOGGED, +VIO), where LOSTEALRAT = y, LOGGED = x_1, VIO = x_2, and $\delta_1 = \delta_2 = +1$.

FIG. 4: RELATIONSHIP AMONG DEPENDENT VARIABLES IN VMSYS1
(EACH POINT IS A 15 MINUTE INTERVAL)



The approach in example 1 is based on the following observation about monotone functions. Given a monotone function f, $X_1 = (x_{11}, \ldots, x_{1J})$, and $X_2 = (x_{21}, \ldots, x_{2J})$, then $f(X_1) \leq f(X_2)$ if X_1 has a smaller value for each x_j which increases with f and a larger value for each x_j which decreases with f. Thus, we can use f to impose a **partial order** ($\prec_f$):

$X_1 \prec_f X_2 \Leftrightarrow$ for all j, $\delta_j x_{1j} < \delta_j x_{2j}$

$X_1 \prec_f X_2$ implies $f(X_1) \leq f(X_2)$

We emphasize that $\prec_f$ is a partial order; so, not all X_1 and X_2 can be compared. Specifically, **X_1 and X_2 are comparable** if either $X_1 \prec_f X_2$ or $X_2 \prec_f X_1$. In example 1, $X_1 = (482, 486)$ and $X_2 = (510, 611)$ are comparable since $(+1)482 < (+1)510$ and $(+1)486 < (+1)611$, so $X_1 \prec_f X_2$. Furthermore if $X_1 \prec_f X_2$, we say that **(y_1, X_1) and (y_2, X_2) agree with $y(\Delta X)$** if $y_1 \leq y_2$.

Now we can state the general approach suggested by example 1. Suppose we want to bound y^* when the independent variables take on the values X^*. If all pairs of measurement instances agree with $y(\Delta X)$, an excellent lower bound is $\max\{y_l \mid X_l \prec_f X^*\}$ and an excellent upper bound is $\min\{y_h \mid X^* \prec_f X_h\}$.

By itself, the foregoing is inadequate because it does not handle those cases in which comparable X's do not agree with $y(\Delta X)$. We handle this by adding an error term to account for deviations from the monotone relationship.

$$y_i = f(X_i) + \varepsilon_i \qquad (3)$$

Where:

f = monotone function with known directional effects

$\varepsilon_1, \ldots, \varepsilon_I$ = error terms which are independent and identically distributed random variables

Thus if $\varepsilon_i = 0$, the aforementioned procedure is reasonable.

We note that equation (3) has the same form as the regression equation in (2). However, NIMF generalizes on regression in two ways. First, NIMF does not assume a known functional form for f; instead, NIMF only requires knowing the directional effect on y (through f) of increasing x_j (as indicated by δ_j). Secondly, the error terms need not be normally distributed; in fact, NIMF makes no distributional assumptions, so it is a non-parameteric technique. Further, NIMF is fairly robust to violations of these assumptions in that they need hold only for X close to X^*.

To summarize,

Obtaining Prediction Intervals Using NIMF

Given

$y_1, \ldots, y_I$ = dependent variables with $y_I = f(X_I) + \varepsilon_I$

$X_1, \ldots, X_I$ = vectors of independent variables with $X_I = (x_{I1}, \ldots, x_{IJ})$

f = monotone function for which the directional effect of increasing each x_j is known

$\varepsilon_1, \ldots, \varepsilon_I$ = error terms which are independent and identically distributed random variables

$1 - \alpha$ = confidence level

X^* = values of independent variables for an unknown y^*

Find

y_L and y_H such that $Pr[y_L \leq y^* \leq y_H] \geq 1 - \alpha$.

Section 4.1 describes the NIMF technique, and appendix B presents a proof that NIMF constructs valid prediction intervals. Section 4.2 shows how the quality of a NIMF model can be evaluated in a manner similar to how R^2 is used for regression models.

4.1 NIMF Technique

NIMF is a two-step technique with an optional third step. The first step selects a **set of potential lower bounds (S_L)** and a **set of potential upper bounds (S_H)** based on the partial order imposed by f. Specifically, in order for y_l to be in S_L, $X_l \prec_f X^*$; similarly, for y_h to be in S_H, $X^* \prec_f X_h$. This ensures that $f(X_l) \leq f(X^*) \leq f(X_h)$; however, it does not guarantee that $y_l \leq y^* \leq y_h$ since we do not know the values of the error terms in equation (3). Step 2 addresses this issue by employing a non-parametric statistical technique to select a member of S_L as a lower bound for y^* and a member of S_H as an upper bound for y^*. An optional third step ensures that the resulting bounds are consistent with the monotone relationship; this is useful for explaining predictions in terms of the monotone relationship and also provides a form of error checking. (See Fig. 5 for a summary of these steps.)

The 3 steps in the NIMF technique are described below. This description is aided by a continuing example in which the following are given:

Continuing Example for Section 4.1

Bound the unknown y^* based on the following

$y(\Delta X)$ = LOSTEALRAT(+LOGGED, +VIO)
$1-\alpha$ = 75%
X^* = (500, 500)

Table 1: Base Measurement Data for Section 4.1 Continuing Example

Entry #	LOGGED (x_1)	VIO (x_2)	LOSTEALRAT (y)
1	510	320	.5
2	450	680	.4
3	490	720	.7
4	391	350	0
5	457	491	.2
6	466	407	0
7	400	400	.1
8	482	486	.3
9	492	492	.6
10	451	465	1.2
11	509	500	.4
12	530	502	.5
13	531	600	.7
14	510	611	1.5
15	558	522	1.1
16	526	544	1.6
17	524	534	1.4

Step 1: Select the Possible-bounds Sets

We want to pick S_L, a set of potential lower bounds for y^*, and S_H, a set of potential upper bounds for y^*. We can do this by using the partial order imposed by f ($\prec_f$). Specifically, if $X \prec_f X^*$, then $f(X) \leq f(X^*)$. (Of course, we are not assured that $y \leq y^*$ since it may be that $\varepsilon >> \varepsilon^*$.) Let $\mathbf{S}_L^0 = \{y_l \mid X_l \prec_f X^*\}$. We require that $S_L \subseteq S_L^0$. Similarly, $\mathbf{S}_H^0 = \{y_h \mid X^* \prec_f X_h\}$, and $S_H \subseteq S_H^0$.

In our continuing example, $S_L^0 = \{y \mid (y,X)$ in Table 1 , LOGGED < 500, VIO $< 500\}$ and $S_H^0 = \{y \mid (y,X)$ in Table 1 , LOGGED > 500, VIO $> 500\}$, which results in:

S_L^0 :	0	.2	0	.1	.3	.6	1.2
S_H^0 :	.5	.7	1.5	1.1	1.6	1.4	

(Note that entry 11 is not included because $VIO_{11} = 500$.)

How should we choose S_L and S_H? We could let $S_L = S_L^0$ and $S_H = S_H^0$. However, if S_L^0 or S_H^0 contain X very distant from X^*, then $|f(X) - f(X^*)|$ will be large and hence the prediction interval will be wide. This motivates making S_L and S_H as small as possible. However, these sets should not be too small since doing so will make NIMF more sensitive to large values of $|\varepsilon|$ (i.e., outliers) which would again widen the prediction interval. We define M as the maximum size of the potential bounds sets, and for simplicity we require that $size(S_L) = size(S_H)$. We define the size of S_L and S_H to be N, where $N = \min\{M, size(S_L^0), size(S_H^0)\}$. Our studies suggest that $10 \leq M \leq 20$ works well.

If the size of S_L^0 exceeds N, we must remove elements from S_L^0 to construct S_L, and similarly for constructing S_H from S_H^0. How should we choose which elements to remove? To minimize $|f(X) - f(X^*)|$, we keep those elements which are closest to X^*. We define distance as

$$d(X_1, X_2) = \left(\sum_{j=1}^{J} \frac{(x_{1j} - x_{2j})^2}{s_j^2} \right)^{\frac{1}{2}} \tag{4}$$

Where:

s_j^2 = standard deviation of x_j

Then,

$$S_L = \{y_l \in S_L^0 \text{ with the } N \text{ smallest } d(X_l, X^*)\} \tag{5a}$$

$$S_H = \{y_h \in S_H^0 \text{ with the } N \text{ smallest } d(X_h, X^*)\} \tag{5b}$$

In our continuing example, $N = 6$. Since $size(S_L^0) = 7 > 6$, we must delete one element from S_L^0 to form S_L. The element of S_L^0 with the largest $d(X, X^*)$ is entry 4 in Table 1. So,

S_L :	.2	0	.1	.3	.6	1.2
S_H :	.5	.7	1.5	1.1	1.6	1.4

Fig. 5: Summary of NIMF

Problem Statement

Given

$y_1, \ldots, y_I$ = dependent variables with $y_i = f(X_i) + \varepsilon_i$
$X_1, \ldots, X_I$ = vectors of independent variables with $X_i = (x_{i1}, \ldots, x_{iJ})$
f = monotone function for which directional effect of increasing x_j is known
$\varepsilon_1, \ldots, \varepsilon_I$ = error terms which are independent and identically distributed random variables
$1-\alpha$ = confidence level
X^* = values of independent variables for y^*

Find

y_L and y_H such that $Pr[y_L \leq y^* \leq y_H] \geq 1 - \alpha$.

Step 1: Select the Potential-bounds Sets

$S_L = \{y_l \in S_L^0$ with the N smallest $d(X, X^*)\}$
$S_H = \{y_h \in S_H^0$ with the N smallest $d(X, X^*)\}$

Where:

$S_L^0 = \{y_l \mid X_l \prec_f X^*\}$
$S_H^0 = \{y_h \mid X^* \prec_f X_h\}$
$\prec_f$ = partial order imposed by f

$$d(X_1, X_2) = \left(\sum_{j=1}^{J} \frac{(x_{1j} - x_{2j})^2}{s_j^2} \right)^{\frac{1}{2}}$$

s_j^2 = standard deviation of x_j
$N = \min[M, size(S_L^0), size(S_H^0)]$
M = maximum size for S_L and S_H

Step 2: Form An Initial Prediction Interval

1. Sort S_L yielding $y_{l,1} \leq y_{l,2} \leq \cdots \leq y_{l,N}$
   Sort S_H yielding $y_{h,1} \leq y_{h,2} \leq \cdots \leq y_{h,N}$
2. $k = \max\left\{ m \text{ such that } \sum_{n=m}^{N} \binom{N}{n} .5^N \geq 1 - \frac{\alpha}{2} \right\}$
3. $y_L = \min\{y_{l,k}, y_{h,N-k+1}\}$
   $y_H = \max\{y_{l,k}, y_{h,N-k+1}\}$

Step 3 (optional): Find Bounds Consistent with the Monotone Function

$y_L^a = y_L$ from step 2, and $y_H^a = y_H$ from step 2.

$$y_L = \begin{cases} -\infty & \text{if } y_L^a < \min S_L \\ \max\limits_{y_l \in S_L} \{y_l \leq y_L^a\} & \text{otherwise} \end{cases}$$

$$y_H = \begin{cases} +\infty & \text{if } y_H^a > \max S_H \\ \min\limits_{y_h \in S_H} \{y_h \geq y_H^a\} & \text{otherwise} \end{cases}$$

Step 2: Form An Initial Prediction Interval

We obtain a prediction interval for y^* by applying a nonparametric statistical technique. We first sort S_L and re-index its elements so that $y_{l,1} \leq \cdots \leq y_{l,N}$. Similarly, S_H is sorted and re-indexed so that $y_{h,1} \leq \cdots \leq y_{h,N}$. Doing so in our continuing example, results in the following:

Index:	1	2	3	4	5	6
S_L :	0	.1	.2	.3	.6	1.2
S_H :	.5	.7	1.1	1.4	1.5	1.6

A $1-\alpha$ prediction interval can then be obtained by selecting a lower bound from S_L and an upper bound from S_H using a variation on a technique for obtaining confidence intervals for the median of a distribution [Bradley 68]. Our technique computes an index k so that $y_{l,k}$ will be the lower bound and $y_{h,N-k+1}$ will be the upper bound. The index is computed as follows:

$$k = \max\left\{ m \text{ such that } \sum_{n=m}^{N} \binom{N}{n} .5^N \geq 1 - \frac{\alpha}{2} \right\}$$

For those cases where there is no such k, $y_L = -\infty$ and $y_H = +\infty$. Referring to our continuing example, $N = 6$ and $1 - \frac{\alpha}{2} = .88$

$$\sum_{n=1}^{6} \binom{6}{n} .5^6 = .98 > .88 \qquad (m = 1)$$

$$\sum_{n=2}^{6} \binom{6}{n} .5^6 = .89 > .88 \qquad (m = 2)$$

$$\sum_{n=3}^{6} \binom{6}{n} .5^6 = .64 < .88 \qquad (m = 3)$$

So, $k = \max\{1, 2\} = 2$ and $N - k + 1 = 5$. Thus, $Pr[.1 = y_{l,2} \le y^* \le y_{h,5} = 1.5] \ge .75$.

In general, we are not assured that $y_{l,k} \le y_{h,N-k+1}$. Thus, we define the prediction interval bounds as follows:

$$y_L = \min\{y_{l,k},\ y_{h,N-k+1}\} \tag{6a}$$

$$y_H = \max\{y_{l,k},\ y_{h,N-k+1}\} \tag{6b}$$

In our continuing example,

$$y_L = \min\{y_{1,2},\ y_{h,5}\} = .1$$

$$y_H = \max\{y_{1,2},\ y_{h,5}\} = 1.5$$

Step 3 (Optional): Find Bounds Consistent with the Monotone Relationship

While step 2 produces a prediction interval, we may want to ensure that the prediction interval is consistent with the monotone relationship in the sense that $y_L \in S_L$ and $y_H \in S_H$. This is desirable for two reasons. First, it makes explaining the predictions much easier. For instance, we can explain the lower bound in our continuing example as follows:

> Entry 7 in table 1 (i.e., $\text{LOSTEALRAT}_7 = .1$) is a lower bound for LOSTEALRAT^* since:
>
> 1. LOSTEALRAT increases with LOGGED and VIO and
> 2. $\text{LOGGED}_7 = 400 < 500 = \text{LOGGED}^*$ and $\text{VIO}_7 = 400 < 500 = \text{VIO}^*$

Secondly, a form of error checking is provided for those cases in which $y_L < \min S_L$ or $y_H > \max S_H$. Such situations indicate a potential violation of the monotone relationship which can impair the accuracy of the predictions. Under such circumstances, it is better to give no bound (i.e., ∞) rather than one based on a faulty assumption.

Let y_L^a be y_L from step 2, and y_H^a be y_H from step 2. To ensure bounds consistent with the monotone relationship, step 3 chooses y_L to be the largest member of S_L such that $y_l \le y_L^a$. Similarly, y_H is the smallest member of S_H such that $y_h \ge y_H^a$. Note that the confidence level remains at least $1 - \alpha$, although the width of the prediction interval may increase. More formally,

$$y_L = \begin{cases} -\infty & \text{if } y_L^a < \min S_L \\ \max\limits_{y_l \in S_L} \{y_l \le y_L^a\} & \text{otherwise} \end{cases} \tag{7a}$$

$$y_H = \begin{cases} +\infty & \text{if } y_H^a > \max S_H \\ \min\limits_{y_h \in S_H} \{y_h \ge y_H^a\} & \text{otherwise} \end{cases} \tag{7b}$$

In our continuing example, we have

$$y_L = \max_{y_1 \in S_L} \{y_1 \le y_{1,2} = .1\} = .1$$

$$y_H = \min_{y_h \in S_H} \{y_h \ge y_{h,5} = 1.5\} = 1.5$$

4.2 Evaluating A NIMF Model

A NIMF model is simply a monotone relationship denoted by $y(\Delta X)$ which specifies a set of independent variables (X) and the directional effect on the dependent variable (y) when an independent variable increases. The accuracy of NIMF predictions can be strongly affected by the degree to which $y(\Delta X)$ holds. Here we present a simple metric, **A**, which assesses the degree to which $y(\Delta X)$ holds for a specific set of measurement data.

One approach to assessing $y(\Delta X)$ is to look at the **fraction of comparable measurement-pairs which agree with** $y(\Delta X)$ (denoted by F_A). That is, given pairs of measurement instances of the form $((y_1, X_1), (y_2, X_2))$ with $X_1 <_r X_2$, for how many such pairs is it true that $y_1 \le y_2$? While F_A has much appeal, it is a little deceiving since we expect some level of agreement just due to chance. Intuitively, we expect that for an arbitrarily selected monotone relationship, a comparable measurement-pair will be equally as likely to agree with the monotone relationship as to disagree with it. So, we assess $y(\Delta X)$ by the increase in agreement compared to what we expect due to chance. Or,

$$A = \frac{F_A - .5}{.5} \tag{8}$$

Where:
- A = increase in agreement compared to chance
- F_A = fraction of comparable measurement-pairs which agree with $y(\Delta X)$

In essence, A assesses how well $y(\Delta X)$ explains the measurement data. If agreement is purely due to chance, $A = 0$. If 100% of the measurement-pairs agree with $y(\Delta X)$, then $F_A = 1$, so $A = 1$. We have found A to be an effective way of evaluating NIMF models.

5. Case Studies

This section compares the accuracy of several least squares regression models to the accuracy of NIMF for predicting contention for the first 16 megabytes of storage in three VM SP/HPO 4.2 systems. (See Fig. 1 for a description of these systems.) Regression and NIMF are compared using the same base measurement data and the same independent variables (i.e., LOGGED and VIO). Models are compared in two ways:

1. Self-evaluation
   How well does the model explain the base measurement data (e.g., R^2 for regression and A for NIMF)?
2. Cross validation
   How accurately can the model predict **cross validation data** (measurements not present in the base measurement data)? There are two criteria:
   a. the **empirical confidence level** which is the fraction of the prediction intervals which contain the actual value (from the cross-validation data)
   b. the average prediction interval width.

For item 1, we want large values for R^2 and A. Item 2 specifies two criteria which we combine into a single goal: minimize the average prediction interval width subject to the constraint that the empirical confidence level is at least $1 - \alpha$. Herein, $1 - \alpha = .75$, so we want the empirical confidence level to be at least 75%.

Throughout these studies, we set M, the maximum size of S_L and S_H, to 20 and use the NIMF model below:

$$y(+\text{LOGGED}, +\text{VIO}) \tag{9}$$

Where:
$y \in \{\text{LOSTEALRAT, FREXTND, PRIMEUTIL}\}$
Also, we use step 3 of the NIMF procedure to obtain bounds consistent with the monotone relationship; so the NIMF prediction intervals tend to be wider than if step 3 is not used.

The choice of the regression model is more difficult because we have little intuition about the algebraic relationship between the independent and dependent variables. So, we take the same approach as in section 3. (See R1, R2, and R3 where y may be LOSTEALRAT, FREXTND, or PRIMEUTIL.) We consider polynomials of increasingly higher degree until either: (a) there is little increase in R^2 or (b) the average width of the regression prediction intervals is smaller than the average width of the NIMF prediction interval.

We begin with LOSTEALRAT in VMSYS1. Recall from section 3 that R3 produces the best R^2 (.37). Fig. 6a shows the regression prediction intervals and the cross validation data for a 75% confidence level. The empirical confidence level is certainly adequate - 94%. However, the average width of the prediction intervals is 1.25, which seems fairly wide. Indeed, in several cases the R3 prediction interval includes negative values, which is unreasonable since LOSTEALRAT is a rate.

Fig. 6b plots 75% NIMF prediction intervals for the same VMSYS1 base measurement data and cross validation data. Again the empirical confidence level (83%) is acceptable. Furthermore, the average prediction interval width is .54 which is 57% smaller than the R3 prediction interval. Also note that the prediction intervals do not contain negative values. Indeed, NIMF always produces prediction intervals with feasible values for the bounds since NIMF uses actual measurement data for the bounds.

Fig. 6: Regression vs. NIMF for VMSYS1 LOSTEALRAT
y = LOSTEALRAT, L = LOGGED, V = VIO
PI = prediction interval

Fig. 6a: Regression Model

$$y = b_0 + b_1L + b_2V + b_3L^2 + b_4LV + b_5V^2 + b_6L^3 + b_7L^2V + b_8LV^2 + b_9V^3 + \varepsilon$$

75% PI % in PI = 94 Avg. Width = 1.25 R^2 = .37

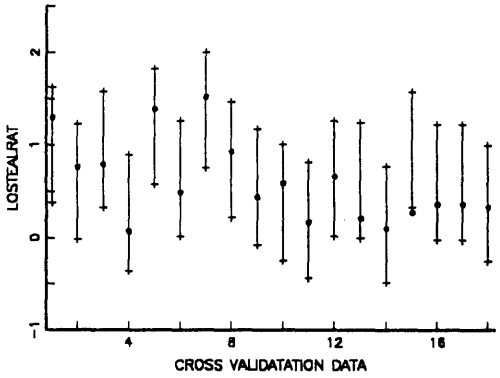
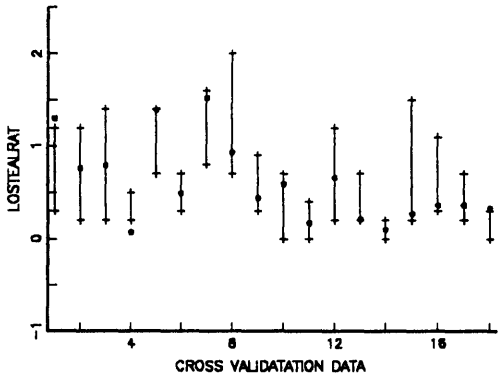


Fig. 6b: NIMF Model

$y(+L, +V)$

75% PI % in PI = 83 Avg. Width = .54 A = .58



Why does NIMF produce prediction intervals which are so much narrower than R3? One explanation is that R3 is probably a poor approximation to the 'true' function relating LOGGED and VIO to LOSTEALRAT. Such a situation results in equational bias which makes the error terms in the regression equation larger and thereby causes wider prediction intervals. Since NIMF does not rely on an explicit form for the function relating LOGGED and VIO to LOSTEALRAT, NIMF does not suffer from equational bias.

Another explanation relates to the nature of the VMSYS1 base measurement data. Recall that regression assumes that the dependent variable has a constant variance over the entire range of independent variables; regression prediction intervals are not very robust to violations of this assumption. Note that the variance of LOSTEALRAT in VMSYS1 increases with LOGGED and VIO. Thus, the overall standard error is too large for small values of LOGGED and VIO, and the cross validation data are mostly in this range; a large standard error creates a wide prediction interval. While NIMF also assumes a constant variance (i.e., the error terms are identically distributed), NIMF is robust to violations of this assumption since it need only hold locally around y^*. That is, NIMF looks at the local behavior around y^* by using the monotone relationship to select a set of potential lower bounds (S_L) and potential upper bounds (S_H). So, the assumption of constant variance need only hold within $S_L \cup S_H$. Thus, when the variability within $S_L \cup S_H$ is small, NIMF produces narrower prediction intervals than when the variability within $S_L \cup S_H$ is large.

The preceding study is somewhat of a worst case for regression. How does NIMF compare to regression when regression should do well? The LOSTEALRAT data for VMSYS3 offer an opportunity to answer this question. From Fig. 7, we see that LOSTEALRAT is almost linear in VIO with a somewhat smaller effect due to LOGGED. As before, NIMF, R1, R2, and R3 are used to construct prediction intervals for the cross-validation data. These results are the first set of entries in Fig. 8. In all cases, the empirical confidence level is acceptable. Also, NIMF produces smaller prediction intervals than both R1 and R2. Although NIMF produces wider prediction intervals than R3, they are only 8% wider. Thus, even in a case where regression should do well NIMF is still competitive.

Fig. 8 summaries all of the case studies. Note that for the PRIMEUTIL study on VMSYS1, there is one cross-validation point for which NIMF does not produce bounds. The reason for this is that the monotone relationship does not hold near that X^*, and so step 3 of NIMF sets the bounds to ∞.

We now review all five case studies summarized in Fig. 8. Note that with the exception of R3 for VMSYS3, NIMF produces narrower prediction intervals than all three regression models while still ensuring an acceptable empirical confidence level. It is possible that with additional effort better regression models could be obtained. However, for relatively little effort, NIMF produces surprisingly accurate predictions.

6. Evaluation of NIMF

A predictive model can greatly assist with workload management by anticipating the impact of increasing or decreasing workload. Such a model should have two characteristics:

1. Accuracy
   a. For a $1-\alpha$ prediction interval (e.g., 75%), the empirical confidence level (i.e., the fraction of the actual values which lie within their prediction interval) should be at least $1-\alpha$.
   b. The prediction intervals should be narrow.
2. Ease of model formulation

Ease of model formulation is desirable since resource constraints may be installation specific (e.g., an installation-developed service machine). Thus, if a predictive model is to be used, it must be developed by the installation's performance analysts. Since performance analysts often have many other system programming responsibilities, model construction has to be fast and effective.

NIMF offers advantages in both accuracy and ease of model formulation. The case studies suggest that **NIMF is as accurate as a good regression model.** Specifically, in four of the five case studies, NIMF produces prediction intervals which are 30% to 57% narrower than our best regression model; in the fifth case, the NIMF prediction interval is only 8% wider than that which our best regression model produces. In all cases, the NIMF empirical confidence level is acceptable.

Another aspect of NIMF's accuracy is that its choice of prediction interval bounds tends to be more meaningful than simplistic regression models. For example, all three regression models produce

FIG. 7: LOSTEALRAT VS. INDEPENDENT VARIABLES FOR VMSYS3
(EACH POINT IS A 15 MINUTE INTERVAL)

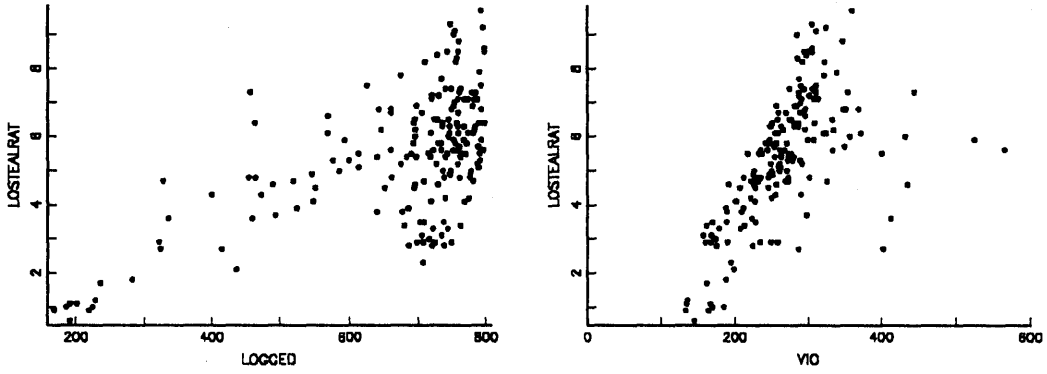


Fig. 8: Summary of Comparisons Between NIMF and Regression Cross Validation Data with 75% Prediction Intervals

System	Dependent Variable	Method	Self-Eval	% in PI	% Chg. in PI Width Compared to NIMF
VMSYS3	LOSTEALRAT	NIMF	$\Lambda = .70$	90	
		R1	$R^2 = .47$	100	30
		R2	$R^2 = .66$	90	4
		R3	$R^2 = .74$	80	-8
VMSYS1	LOSTEALRAT	NIMF	$\Lambda = .58$	83	
		R1	$R^2 = .25$	100	148
		R2	$R^2 = .35$	94	133
		R3	$R^2 = .37$	94	131
VMSYS1	FREXTND	NIMF	$\Lambda = .80$	88	
		R1	$R^2 = .32$	88	100
		R2	$R^2 = .44$	94	82
		R3	$R^2 = .45$	94	82
VMSYS1	PRIMEUTIL	NIMF*	$\Lambda = .38$	83	
		R1	$R^2 = .073$	100	49
		R2	$R^2 = .11$	100	46
		R3	$R^2 = .16$	100	43
VMSYS2	FREXTND	NIMF	$\Lambda = .58$	75	
		R1	$R^2 = .21$	100	58
		R2	$R^2 = .22$	100	58

* One NIMF point for prime utilization had no lower bound
PI = prediction interval

prediction intervals with negative values for the low-DPA page steal rate and free storage extend rate; NIMF never does. This characteristic results from NIMF choosing the prediction interval bounds from measurement data. Thus, **NIMF prediction interval bounds will always be feasible values of the dependent variable** since the bounds are taken from actual measurement data.

A further consideration relating to accuracy is robustness to outliers. Since least squares regression is a global technique (i.e., all of the base measurement data are used to construct each prediction interval), a few outliers can greatly affect all of the predictions. In fact, an important part of building a regression model is identifying outliers and removing their effects [Daniel and Wood 80]. On the other hand, NIMF is relatively insensitive to outliers since NIMF's assumptions need only hold locally around y^*. This results from NIMF selecting prediction interval bounds based on the potential-bound sets S_L and S_H. Specifically, consider the case where $size(S_L) = size(S_H) = 20$. A prediction interval for y^* is affected only by an outlier which is one of the 20 closest points less than X^* or one of the 20 closest points greater than X^*. If we further assume a 75% confidence level, then the (initial) prediction interval bounds will be the seventh smallest element in S_L and the seventh largest element in S_H. Thus, there would have to be at least seven outliers in S_L or in S_H in order for the prediction interval to be seriously affected.

NIMF also has significant advantages for model formulation. Formulating a NIMF model only requires specifying a monotone relationship between the independent and dependent variables. In contrast, least squares regression requires specifying an algebraic relationship. Finding a good algebraic relationship can be extremely difficult, as indicated by our attempts in section 3 to construct a regression model for the low-DPA page steal rate and by the case studies in section 5. In theory, a polynomial can approximate any 'smooth' function. Indeed, our approach to the regression models has been to use polynomials of increasingly higher degree. However, our experience indicates that much more sophistication is required. Instead of trying to find a good algebraic expression relating contention for the first 16 megabytes of storage to workload variables, we use NIMF to exploit a fairly obvious monotone relationship - contention for the first 16 megabytes of storage should increase with workload. Our experience has been that **NIMF can make model formulation trivial.**

NIMF does have two shortcomings. First, unlike regression, NIMF is inherently an interpolative technique. Thus, predictions can be made only in variable regions where there are measurement data. For workload management, this may not be a severe drawback since installations routinely collect large volumes of measurement data over a wide range of variable values.

A second issue with NIMF arises from its requiring a monotone relationship between dependent and independent variables. While monotone relationships are quite general and are easy to state, not all functions are monotone. One technique for handling non-monotone functions is to partition them into several monotone sub-functions. We are currently pursuing this idea.

An additional consideration when using NIMF is that its CPU requirements can be greater than those for regression. In regression, all of the base measurement data are examined when fitting the regression equation. However, each prediction only requires a few algebraic operations. For example in regression model R1, each point estimate requires 2 multiplications and 2 additions. On the other hand, NIMF examines all of the base measurement data for each prediction (so as to compute the potential-bounds sets). We have used NIMF with base measurement data ranging in size from 100 to 500 entries and have consistently obtained execution times for NIMF which are comparable to those for regression. However, NIMF's processing requirements may become an issue if more base measurement data are used.

Our overall feeling is that NIMF is a useful technique if sufficient data are present and monotonicity is a reasonable assumption. Indeed, our experience with NIMF extends beyond predicting contention for the first 16 megabytes of storage in VM SP/HPO. We have used NIMF to develop high-level models for a variety of MVS subcomponents (e.g., the input/output supervisor and the auxiliary storage manager). As with contention for the first 16 megabytes of storage in VM SP/HPO, we have found these models easy to formulate and their accuracy to be comparable to a good regression model.

7. Application of NIMF to Workload Management

Two questions asked in workload management are:

1. How much can workload be increased before resource utilizations are excessive?
2. How much must workload be decreased to obtain acceptable resource utilizations?

An effective way to answer these questions is to use a predictive model to plot the percentage change in workload against measures of resource contention. This section uses the NIMF model LOSTEALRAT(+LOGGED, +VIO) to explore the effect on the VMSYS1 low-DPA page steal rate (LOSTEALRAT) when we vary the number of logged-on users (LOGGED) and the total virtual machine VIO rate (VIO).

In the base measurement data for VMSYS1, the average value of LOGGED is 500 and for VIO is 500. We take these values as the initial workload; that is, the workload values for a 0% change are LOGGED = 500 and VIO = 500. Similarly, a 10% increase in total workload results in LOGGED = 550 and VIO = 550. We also consider the impact of only changing LOGGED while VIO = 500 and only changing VIO while LOGGED = 500.

Fig. 9 plots the results. Note that when both LOGGED and VIO increase, LOSTEALRAT grows quickly. Indeed, a 12% increase in both workload variables results in a low-DPA page steal rate between .8 and 1.6, which might be of concern [Hoag 86] Given the plots in Fig. 9, what can be done? One possibility arises from observing that LOGGED has a much greater impact on LOSTEALRAT than does VIO. Furthermore, in VMSYS1 a large fraction of the logged-on users are idle for an extended period of time. This suggests that LOSTEALRAT might be controlled by logging-off such users. Of course the benefits of doing so must be weighed against the inconvenience caused to users.

8. Conclusions

Workload management can be made easier by being able to predict resource contention from workload variables. In large VM SP/HPO systems, contention for the first 16 megabytes of storage is a major performance concern. Since we know of no existing model to predict this type of storage contention, we have attempted to develop such a model.

Our attempts to use least squares regression met with little success because regression requires us to provide something that we do not know - an algebraic expression relating workload to contention for the first 16 megabytes of storage. However, the regression studies do suggest that contention for the first 16 megabytes of storage is monotonically increasing in the workload variables.

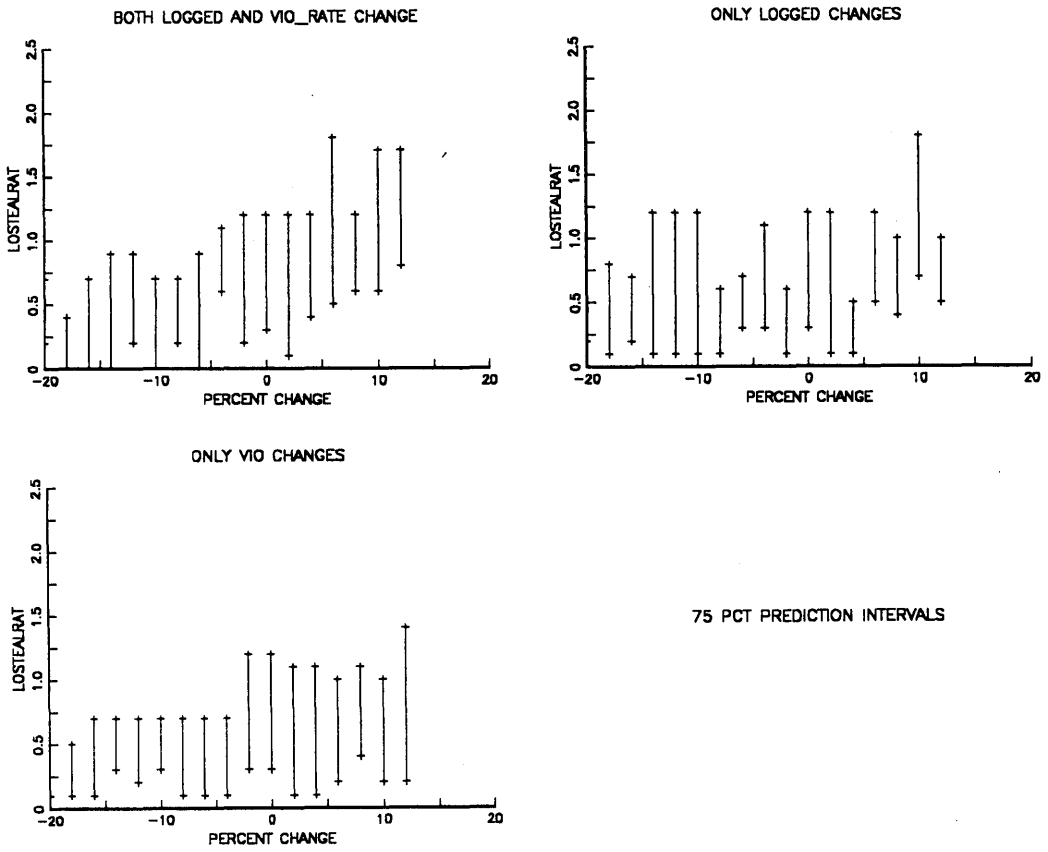
This observation has motivated the development of an apparently new technique - non-parameteric interpolative-forecasting for monotone functions (NIMF). NIMF is a generalization of a technique commonly used by performance analysts. A NIMF model is a monotone relationship which specifies the directional effect on the dependent variable (i.e., increases or decreases) when an independent variable increases. NIMF constructs a prediction interval by using the monotone relationship to select the prediction interval's lower and upper bounds from the measurement data.

Our experience suggests that NIMF is accurate and NIMF models are easy to formulate. Case studies of three VM SP/HPO 4.2 systems suggest that NIMF is as accurate as a good regression model. Specifically, in four of the five case studies, NIMF produces prediction intervals 30% to 57% narrower than the best regression model considered. The fifth case is one in which we expect regression to do well. Even so, NIMF produces narrower prediction intervals than two of the regression models, and NIMF's prediction intervals average only 8% wider than the third regression model. In all cases, the empirical confidence level (i.e., the fraction of the actual values which lie in their prediction interval) equals or exceeds the expected confidence level (i.e., 75%). Additionally, NIMF always selects feasible values for the prediction interval bounds, unlike regression models. For example in the case studies, the regression prediction intervals include negative values for low-DPA page steal rate. Furthermore, NIMF can be made fairly robust to outliers by controlling the maximum size of the potential-bounds sets.

NIMF facilitates model formulation by only requiring models to specify the directional effect on the dependent variable when an independent variable increases. In the case of contention for the first 16 megabytes of storage, this has tremendous advantages since we have little intuition about its algebraic relationship to workload.

NIMF has two limitations: (1) it is inherently an interpolative technique, and (2) the function relating the independent variables to the dependent variable must be monotone. The former may not be a problem for workload management since installations routinely collect large volumes of measurement data over a wide range of variable values. The latter can be a more serious limitation; however, when monotonicity is a reasonable assumption, NIMF can be a

Fig. 9: NIMF Predictions of Effect of Workload Changes in VMSYS1



powerful technique. An additional consideration when using NIMF is that it can impose greater processing requirements than regression; however, we have not found NIMF's processing requirements to be excessive.

Although our experience with NIMF is limited, we feel that it can be an important tool for managing computer performance. For example, we proposed a strategy for controlling low-DPA page steals in a large VM SP/HPO system by using NIMF to predict the effect of workload changes on contention for the first 16 megabytes of storage. More generally, performance management often requires considering installation-specific resource constraints (e.g., an installation-developed service machines) which are difficult to take into account in pre-packaged models. One could construct a queueing or regression model, but these techniques require specifying algebraic relationships which may be non-trivial to discover. NIMF offers an appealing alternative since one need only specify directional effects rather than algebraic relationships.

Acknowledgements

We wish to thank Thomas Lane for statistical consultations, Margaret Doherty and Tennie Eisler for their assistance with VMMAP data, Norman Waite for his assistance with the formal validation of NIMF, and Yonathan Bard, Robert Berry, Jonathan Hosking, Stephen Lavenberg, Keith Milliken, Randolph Nelson, Thomas Philips, Betty Schultz, William White, Hugo Van Woerkom, and Emmanuel Yashchin for their thoughtful comments.

References

[Daniel and Wood 80] Cuthbert Daniel and Fred S. Wood, **Fitting Equations to Data,** John Wiley & Sons, New York, 1980.

[de Boor 78] Carl de Boor, **A Practical Guide to Splines,** Springer-Verlag, New York, 1978.

[BGS 84] BGS Systems Inc., *An Introduction to VM Capacity Management,* Document NO: 83035-01, 1984.

[Bradley 68] James Bradley, **Distribution-Free Statistical Tests,** Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1968.

[Denning and Buzen 78] Peter J. Denning and Jeffrey P. Buzen, *The Operational Analysis of Queueing Network Models,* Computer Surveys, Vol. 10, No. 2, September 1978, pp. 225-261.

[Draper and Smith 68] N. R. Draper and H. Smith, **Applied Regression Analysis,** John Wiley & Sons, Inc., New York, 1966.

[Hoag 86] D. C. Hoag, *Managing VM/SP HPO Real Storage Constraint Below the 16 Megabyte Line,* Washington Systems Center Technical Bulletin, GG66-0261, 1986.

[IBM 84] IBM Corp., *VMPPF Guide and Reference,* SC34-2128-1, 1984.

[IBM 85] IBM Corp., *Virtual Machine Monitor Analysis Program: User's Guide and Reference,* SC34-2166-1, 1985.

[Kleinrock 75] Leonard Kleinrock, **Queueing Systems Volume I: Theory and Applications,** John Wiley & Sons, New York, 1975.

[Schatzoff and Lane 87] Martin Schatzoff and Thomas Lane, *Regression Analysis in Grafstat,* IBM Research Report RC11654, 1986.

Appendix A: List of Symbols

Λ = fractional increase in agreement with the monotone relationship compared to that expected due to chance; assesses how well a monotone relationship fits the data
$1 - \alpha$ = confidence level for a prediction interval
$d(X_1, X_2)$ = distance between X_1 and X_2
δ_j = directional effect of j-th independent variable; $\delta_j \in \{1, -1\}$
$\Delta = (\delta_1, \ldots, \delta_J)$
$\frac{\partial f}{\partial x_j}$ = first derivative of f with respect to x_j (i.e., directional effect on f of increasing x_j).
h = index of element in S_H or $S_H^{\wp}$
i = index used for measurement instances (observations)
I = number of measurement instances (observations)
j = index used for independent variables
J = number of independent variables
k = index of bounds obtained in step 2 of NIMF
l = index of element in S_L or $S_L^{\wp}$
N = maximum size of a potential bound set
$Pr[y_L \leq y^* \leq y_H]$ = probability of y^* being between y_L and y_H
$S_H^{\wp}$ = set of all potential upper bounds in S
$S_L^{\wp}$ = set of all potential lower bounds in S
S_H = subset of $S_H^{\wp}$ used to construct prediction interval
S_L = subset of $S_L^{\wp}$ used to construct prediction interval
x_j = j-th independent variable
x_{ij} = value of the j-th independent variable for the i-th measurement instance
X = vector of independent variables, $(x_1, x_2, \ldots, x_J)$
X^* = values of independent variables used to predict an unknown y^*
X_i = vector of values of independent variables for the i-th measurement instance, $(x_{i1}, x_{i2}, \ldots, x_{iJ})$
y = dependent variable
y^* = value of dependent variable associated with the independent variables X^*
y_i = value of the dependent variable for the i-th measurement instance
y_h = element in S_H
$y_{h,k}$ = k-th smallest element in S_H
y_H = upper bound for y^*
$y_H^{\wp}$ = upper bound for y^* produced in NIMF step 2
y_l = element in S_L
$y_{l,k}$ = k-th smallest element in S_L
y_L = lower bound for y^*
$y_L^{\wp}$ = lower bound for y^* produced in NIMF step 2
(y_i, X_i) = measurement instance
$y(\Delta X)$ = monotone relationship between X and y; Δ specifies the directional effect on y of each independent variable in X
$\prec_f$ = partial order imposed by f; $X_1 \prec_f X_2 \Leftrightarrow$ for all j, $\delta_j x_{1j} < \delta_j x_{2j}$

A Simple Linear Programming Model For Computation Center Investment Decisions

Matt J. Merges
AT&T Bell Laboratories

ABSTRACT

Computation center management is often faced with the problem of selecting from a set of technically feasible solutions to satisfy a service requirement. In the ideal case, the decision is based on the net present cost of the alternatives. The major problem involved in this approach is the estimation of the cash flows associated with each option. Future cash flows such as salvage values or technical support costs pose major difficulties. As a result, decisions are often made using uncertain information.

A linear programming model is developed to assist the manager in the decision making process. The model selects the least cost alternative available from the set of technically feasible options. The major advantage of the model over existing decision making tools is the sensitivity analysis that results from the solution of the model. The analysis indicates which financial assumptions are critical to the decision and which assumptions are of minor consequence. This information can then be used to determine the degree of certainty of the decision. It also indicates which assumptions need to be refined if a more certain decision is desired.

1. Introduction

Computation center management is often faced with the problem of deciding among a set of technically feasible options to satisfy a service requirement. An example of such a decision is the choice of a computing system to satisfy a projected customer service demand. Several types of new hardware could probably be acquired to meet the expected demand. Existing systems within the computation center might be vacated and reused. Another possibility is using surplus equipment available from other computation centers within the corporation.

In the ideal case the manager should first enumerate the set of technically feasible solutions. Next for each of these options (s)he should calculate the net present value of the cash flows [1] involved over the expected life of the service requirement. Finally, the manager should select the option which has the minimum net present cash flow. This process is illustrated in Figure 1.

The major financial problem we encounter with this methodology is the estimation of the cash flows associated with each feasible option. Quantities such as salvage values or future maintenance costs are difficult to predict. In addition, usually a large number of cash flows are associated with each option.

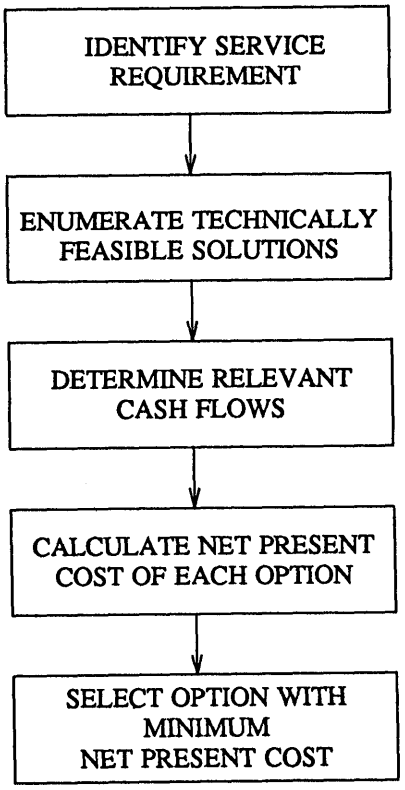

IDENTIFY SERVICE REQUIREMENT
ENUMERATE TECHNICALLY FEASIBLE SOLUTIONS
DETERMINE RELEVANT CASH FLOWS
CALCULATE NET PRESENT COST OF EACH OPTION
SELECT OPTION WITH MINIMUM NET PRESENT COST

Figure 1. Standard Decision Process

Cash flow estimation is widely recognized as an imperfect science requiring major effort. Since we are dealing with incomplete information (estimates), we need to establish degrees of criticality[1] for each of our estimates. To know the accuracy of the information is not sufficient. If we know which of our estimates are critical to the decision and which are of minor consequence, we can channel our limited managerial resources correctly. If a particular estimate is shown to be of significant consequence to the decision, then we want to refine the estimate until we reach an appropriate degree of certainty. [2] If, on the other hand, an estimate is shown to be of little consequence to the decision, then we need not expand any additional resources to refine the information. This method of decision making is illustrated in Figure 2.

Consider the following simple example as an illustration. Suppose we make a decision where one of the cash flow estimates is a salvage value of $100,000. If we know that the salvage value estimate could change by $50,000 without affecting the decision, we probably would be satisfied with the $100,000 number. That is, we would be reasonably certain that our decision will remain valid even though there may be fluctuations from our estimate when the decision is executed. If, on the other hand, we know that a $5,000 change would alter our decision, then we may want to refine our estimate of the salvage value depending on our current level of certainty.

The model developed in this paper determines the least cost alternative from a set of technically feasible options and, as part of the solution, automatically provides a sensitivity analysis of the basic financial assumptions. It is the latter information that is the major value of the model. Formulating an acquisition problem in terms of this model will force a manager to identify the basic financial assumptions involved in the decision. It will also provide the manager with the information required to translate these assumptions into cash flows.

The output from the model can be obtained using other methods that employ break-even financial analysis techniques. [3] The advantage of this model is that the output is comprehensive[2] and that the formulation develops understanding of the underlying cash flow aspects of the decision.

1. We need to understand how critical the estimate is to the decision rather than understanding the absolute accuracy of the estimate. If, for example, we overestimate a cash flow by 100% but, because of other more critical factors it doesn't affect the decision, our estimate is satisfactory from the decision task point of view.

2. Some Basic Linear Programming Concepts

The model we develop in this paper is a simple linear programming model. Before we present the derivation of the model, we discuss a few basic linear programming concepts. These concepts prove the correctness of the model and indicate how to use the output of the model for the desired sensitivity analysis. They will also be of value when we describe the process of translating a specific acquisition decision task into the model.

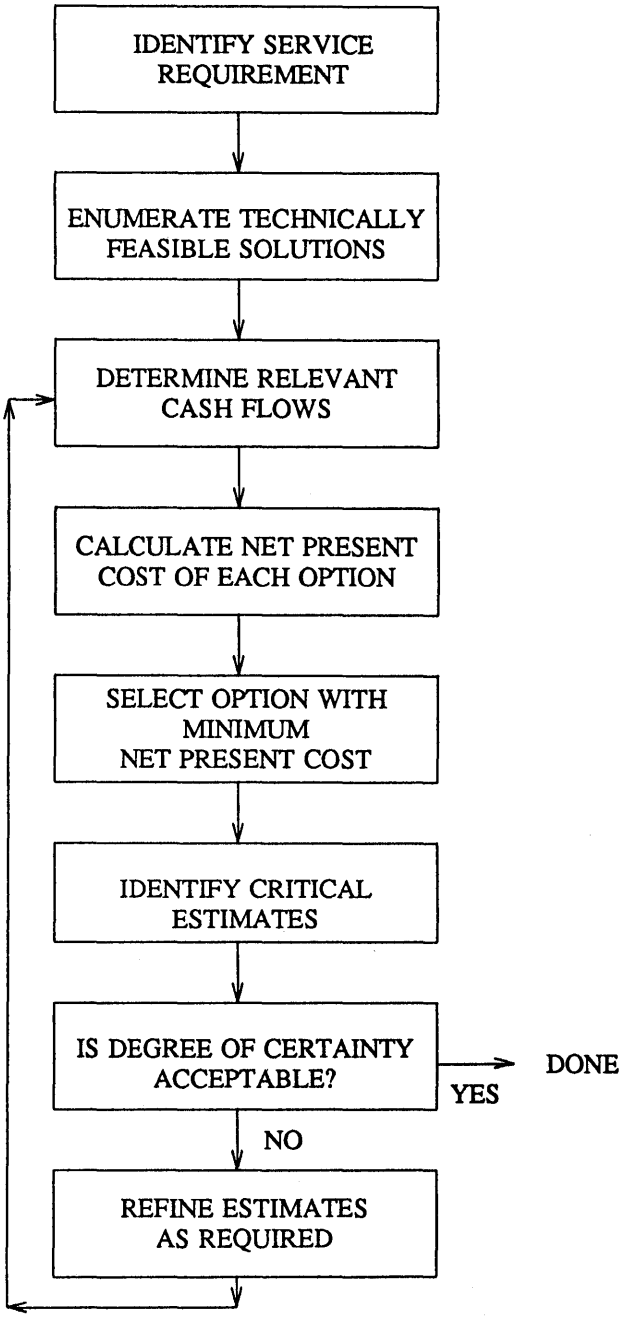


Figure 2. Decision Process with Feedback

2. The model's output automatically provides decision sensitivity information for each of the basic economic assumptions.

2.1 *Format of the Basic Linear Programming Model*

The first concept we introduce is the generic form of the linear programming [4] that is the foundation for our model. We write this linear programming as

$$\textit{minimize} \sum_{i=1}^{m} \sum_{j=1}^{n_i} f_{ij} \times X_{ij} \qquad (2.1)$$

subject to

$$X_{ij} - a_{ij} \times D_i = 0 \qquad \forall\, i \text{ and } \forall\, j \;. \qquad (2.2)$$

In this linear program the objective function, $\sum_{i=1}^{m} \sum_{j=1}^{n_i} f_{ij} \times X_{ij}$, consists of a set of decision variables, $\{X_{ij}\}$, and a set of constants, $\{f_{ij}\}$. These constants are usually called the coefficients of the objective function. The constraints of the program, $\{X_{ij} - a_{ij} \times D_i = 0\}$, introduce an additional set of constants, $\{a_{ij}\}$, and another set of decision variables, $\{D_i\}$.

The two sets of constants are used to define the cost estimates associated with the acquisition decision. They are precisely defined in a section 3 of the paper. The variable sets are used to output the minimum cost option and the sensitivity analysis for the basic estimates.

In this paper, we are not interested in the theory or the techniques that are used to solve linear programming problems. The solution theory is well developed, and many software packages are available for solving the problems. This model generates a very simple class of linear programming problems. They can easily be solved on a desktop computer such as the AT&T PC6300 running the LINDO linear programming package. [5] The examples presented in this paper were solved on a PC6300 with the LINDO system.

2.2 *Stable Coefficient Range*

We will call the set of non zero objective function decision variables that form the solution to one of our linear programming problems the *character* of the solution [6]. That is,

$$X_{ij} \neq 0 \longleftrightarrow X_{ij} \in \textit{character} \;. \qquad (2.3)$$

Now, part of the output from all standard linear programming packages is the range over which a coefficient in the objective function may vary without changing the character of the solution. That is, for each coefficient f_{ij} in the objective function, a range $[L_{ij}, U_{ij}]$ exists such that if f_{ij} is varied unilaterally[3] within this range no change occurs in the character of the solution.

3. Unilaterally in this context means that there are no other changes to the constant sets that define the specific linear programming problem.

This fact is critical to the model's sensitivity analysis. Suppose, for example, that a coefficient in the objective function represents our estimate of the salvage value for a system. The stable range for this coefficient would indicate the range over which the the salvage value could vary without changing the character of the solution. As we see in the following sections the character of the solution will change if and only if our acquisition decision should change. Thus, this range tells us how sensitive our decision is to the estimate of the salvage value.

2.3 *Variable Reduced Cost*

The reduced cost concept applies to objective function decision variables that are not part of the character of the solution. Reduced costs are the amount that the optimum solution would be degraded if a unit of the associated variable is forced into the character. For our model reduced costs are needed to calculate the costs of the options that are not optimum.

3. *Basic Model Formulation*

Computation center management often has multiple options to choose from when deciding how to satisfy a service requirement. For example, if a request is made for a system test machine, they might meet the requirement by acquiring a new system, by using a system scheduled for disposal, or by allocating a portion of an existing system for testing. As we have indicated above, given that each of the options is technically and environmentally feasible, the choice should be based on a present value cost analysis over the expected life of the requirement.

The cost of each option depends on a number of basic cost assumptions. These assumptions usually represent quantities that are not further divisible during the normal decision making process. Examples of these basic assumptions are purchase prices and monthly lease rates for processors. Others include monthly maintenance and environmental costs. Basic cost assumptions can also represent revenue. An example of a revenue assumption is the monies expected for an existing processor when it is used as a trade-in for a newer piece of equipment. We call the basic cost assumptions associated with an option the *cost factors* of the option. We use the notation f_j to denote the jth factor. With this in mind, we can express the cost of an option as a function of its factors. If an option has n factors we write the cost as

$$\textit{Cost} = O(f_1, f_2, \ldots, f_n) \;. \qquad (3.1)$$

where f_j represents the jth cost factor of the option.

To extend this notation to multiple options, we introduce an option index, i, and add this index as a subscript to the notation for cost factors. If option i has n_i cost factors we write the cost as

$$Cost_i = O_i(f_{i1}, f_{i2}, ..., f_{in_i}) . \quad (3.2)$$

The basic assumption of our model is that the actual cost of an option is a linear function of its factors. For example, if a factor, f_{ij}, is the monthly lease rate for a processor, we assume that the total cost contributed by this factor has the form $a \times f_{ij}$, where a is the amount that the factor contributes to the overall cost of the option. For this example, if we are considering a 36-month lease and are assuming an 8% annual cost of money to the corporation, the amount is calculated as

$$a = 1 + \frac{1}{1+.08/12} + \frac{1}{(1+.08/12)^2} \quad (3.3)$$

$$+ \frac{1}{(1+.08/12)^3} + \cdots + \frac{1}{(1+.08/12)^{35}} .$$

We use the notation a_{ij} to denote the amount that the *jth* factor of the *ith* option contributes to the cost of the *ith* option. Thus, the cost of the *ith* option is written as

$$Cost_i = \sum_{j=1}^{n_i} f_{ij} \times a_{ij} . \quad (3.4)$$

Given this notation, a linear programming model for selecting the least cost option from among a set of m options and providing factor sensitivity analysis is

$$minimize \sum_{i=1}^{m} \sum_{j=1}^{n_i} f_{ij} \times X_{ij} \quad (3.5)$$

subject to

$$X_{ij} - a_{ij} \times D_i = 0 \qquad \forall\, i \text{ and } \forall\, j \quad (3.6)$$

$$\sum_{i=1}^{m} D_i = 1 \quad (3.7)$$

$$D_i \geq 0 \qquad \forall\, i . \quad (3.8)$$

It is worthwhile to say at this point that the X_{ij} dimensionless dummy variables are introduced into the formulation of the problem so that the sensitivity analysis will be performed on the basic factors of each option in the decision problem. We want to understand the sensitivity of our decision to the values of f_{ij} but are not particularly interested in the sensitivity to the product $a_{ij} \times f_{ij}$.

4. Correctness of the Model

First, we demonstrate that the model will select the least cost option, from the options described by the two constant sets $\{a_{ij}\}$ and $\{f_{ij}\}$. The selection will be output via the decision variable set $\{D_i\}$. If option l is the least cost option, the variable D_l will have a value of 1 while all other D_is will have a 0 value.

Consider the objective function $\sum_{i=1}^{m} \sum_{j=1}^{n_i} f_{ij} \times X_{ij}$.

By rewriting the constraints

$$X_{ij} - a_{ij} \times D_i = 0 \quad (4.1)$$

as

$$X_{ij} = a_{ij} \times D_i \quad (4.2)$$

and substituting, we can write the objective function as

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} f_{ij} \times a_{ij} \times D_i . \quad (4.3)$$

Using (3.4), this can be rewritten as

$$\sum_{i=1}^{m} Cost_i \times D_i . \quad (4.4)$$

Now, select l such that

$$Cost_l < Cost_i \qquad \forall\, i \neq l . \quad (4.5)$$

Then for $D_i \geq 0$ we have

$$D_i \times Cost_l < D_i \times Cost_i \qquad \forall\, i \neq l . \quad (4.6)$$

Therefore,

$$\sum_{i=1}^{m} D_i \times Cost_l < \sum_{i=1}^{m} D_i \times Cost_i . \quad (4.7)$$

Factoring, we have

$$Cost_l \times \sum_{i=1}^{m} D_i < \sum_{i=1}^{m} D_i \times Cost_i . \quad (4.8)$$

Since $\sum_{i=1}^{m} D_i = 1$, we obtain

$$Cost_l < \sum_{i=1}^{m} D_i \times Cost_i . \quad (4.9)$$

Now, define the set $\{D'_i\}$ as

$$D'_i = 0 \qquad \forall\, i \neq l \quad (4.10a)$$

$$D'_l = 1 . \quad (4.10b)$$

Since $Cost_l = \sum_{i=1}^{m} D'_i \times Cost_i$, we have from (4.9)

$$\sum_{i=1}^{m} D'_i \times Cost_i < \sum_{i=1}^{m} D_i \times Cost_i . \quad (4.11)$$

Thus, the set $\{D'_i\}$ defines the least cost option and will be output from the model due to the minimization request and the two constraints on the decision variable set $\{D_i\}$.

Next, we demonstrate that the desired factor sensitivity analysis results from the solution of the model. Recall that for each coefficient in the objective function,[4] there exists a range over which the coefficient may vary unilaterally without changing the basis of the solution. In our model, we will use the notation $[L_{ij},U_{ij}]$ to specify the stable range of the coefficient f_{ij}.

Consider the *character* of the solution to our model. By definition of character and using the constraints of our model we have

$$X_{ij} \in character \longleftrightarrow D_i = 1 . \quad (4.12)$$

Likewise, we have

$$X_{ij} \in \overline{character}^{5} \longleftrightarrow D_i = 0 . \quad (4.13)$$

Now, consider a new problem defined by the sets $\{a'_{ij}\}$ and $\{f'_{ij}\}$ where

$$a'_{ij} = a_{ij} \qquad \forall\, i \text{ and } \forall\, j \quad (4.14)$$

and

$$f'_{ij} = f_{ij} \qquad \forall\, i \neq p \text{ and } \forall\, j \neq q \quad (4.15)$$

Let

$$f'_{pq} \in [L_{pq},U_{pq}] . \quad (4.16)$$

Then by definition of stable coefficient range we have

$$X_{ij} = 0 \longleftrightarrow X'_{ij} = 0 \quad (4.17)$$

and

$$X_{ij} \neq 0 \longleftrightarrow X'_{ij} \neq 0 . \quad (4.18)$$

From this, it follows that

$$D_i = 0 \longleftrightarrow D'_i = 0 \quad (4.19)$$

and

$$D_i = 1 \longleftrightarrow D'_i = 1 . \quad (4.20)$$

Thus, if a single cost factor, f_{pq}, varies within its stable coefficient range, the least cost option is the same. This implies that our decision is stable. The range, $[L_{pq},U_{pq}]$, defines the sensitivity of our decision to the basic cost estimate f_{pq}.

Consider the following example. If the factor, f_{ij}, represents the cost of a processor and is \$750,000, and sensitivity analysis states that the stable range for this coefficient is [500000,1000000], then we can conclude that as long as the cost is between \$500,000 and \$1,000,000, our optimum choice does not change. This in effect tells us that our estimate of the cost of the processor is probably not critical to the decision at hand.[6]

Note again, however, that the sensitivity analysis provides for a unilateral study of the cost factors (that is to say that all other factors are assumed constant when one is varied within its stable range).

5. *Repeated Factor Problem*

A problem exists with the sensitivity analysis if a factor is common to more than one option. For example, two options of a decision to purchase a new computer system may involve the trade-in of a particular type of system. The factor would be the actual market sell value, but the amount of the factor in each option could differ, for example, because of different depreciation schedules. This is a typical case where different systems of the identical type were acquired at different times.

The problem with the sensitivity analysis for such a case is that the stable coefficient range presented assumes a unilateral change in the coefficient (factor), market sell value. If the same factor appears in more than one option, then the unilateral analysis fails.

This problem is solved by first observing that the cost of option i, $Cost_i$, is a linear function of any of its factors when the others are held constant. For example, if all factors of option i are held constant except factor j, we can write

$$Cost_i = b_{ij} + a_{ij} \times f_{ij} \quad (5.1)$$

where b_{ij} is a constant with the value

$$b_{ij} = \sum_{i \neq j}^{n_i} a_{ij} \times f_{ij} . \quad (5.2)$$

Suppose that we are deciding between two options, p and q, with cost functions as defined above and common factor f_{pj} and f_{qk}. If we assume that all other factors are constant, we can write the cost functions as

$$Cost_p = b_{pj} + a_{pj} \times f_{pj} \quad (5.3)$$

$$Cost_q = b_{qk} + a_{qk} \times f_{qk} . \quad (5.4)$$

Assume that $Cost_p < Cost_q$.

If $a_{pj} = a_{qk}$, the two lines represented by $Cost_p$ and $Cost_q$ are parallel and there is no break-even point. That is, there is no value of f_{pj} (or f_{qk}) which makes the cost of option q less than the cost of option p.

4. For our model, the coefficients are the cost factors $\{f_{ij}\}$.

5. We use the notation $\overline{character}$ to denote the logical compliment of the *character* with respect to the set $\{X_{ij}\}$.

6. We assume that we can estimate the cost of a processor with an accuracy of at least 33%.

In the case where the two lines are not parallel, the break-even point is the value of f_{pj} where the two cost lines intersect. We can find this value by equating the two functions[7] and solving for f_{pj}. We have

$$f_{pj} = \frac{b_{qk} - b_{pj}}{a_{pj} - a_{qk}} . \quad (5.5)$$

The cost equation constants are calculated as

$$b_{pj} = Cost_p - f_{pj} \times a_{pj} \quad (5.6)$$

and

$$b_{qk} = Cost_q - f_{qk} \times a_{qk} . \quad (5.7)$$

Now $Cost_p$ is output as part of the solution to the linear programming problem. It is the computed minimum value of the linear programming problem.

$Cost_q$ can be calculated from the reduced cost of its factors and the corresponding amounts of its coefficients. The calculation is

$$Cost_q = Cost_p + \sum_{i=1}^{n_q} Reduced\ Cost_{qi} \times a_{qi} . \quad (5.8)$$

The reduced costs are also part of the standard output from linear programming systems.

An alternate method to calculate $Cost_q$ for this case is to artificially force the linear programming to select option q. To do this, set one of the factors of option p to a very high value[8] and rerun the program. The resulting value will then be the value of $Cost_q$.

6. *The Model Is Not a Budget Generation Tool*

The purpose of the linear programming model presented in this paper is to provide a tool for better understanding the effects of assumptions on economic decisions. The specification of the options for evaluation need include only those factors relevant to the decision. For example, if the decision involves the selection of a computing system from a number of alternatives, but each alternative requires the same staff for operation, the cost of staff need not be a factor. This fact is important to point out so that results of the linear programming are interpreted correctly. The value of the linear programming solution is simply a number that gives the cost of the optimum solution relative to the other options. It should not be used to generate budgets.

For the sensitivity analysis, however, the stable ranges presented are in terms of absolute dollars for the associated factors.

7. *A Simple Example*

Suppose that we have identified a service requirement to provide a test system for a four year period. Also assume that we have identified two technically feasible options that will satisfy this requirement. Option 1 is to purchase a new system, system A, and use it exclusively for the test requirement. We assume that system A will be able to provide the required processing power over the four year period. If we exercise this option, we will also dispose of a larger older technology system, system B, that will become available at the start of the study interval.

Option 2 is to provide the test environment on system B. This system is able to provide more processing power than the new test requirement so that we will also be able to move test work from one of our production systems to system B. The former test capacity on the production system can be used to meet an expected increase in demand over the four-year period.

7.1 *Option 1 Factors and Amounts*

A. "Cost of System A (f_{11})"

The first factor for option 1 is the cost of the system. Our estimate of the cost is $735,000. The amount of this factor is the initial cost minus depreciation tax benefits. In present value terms[9] the amount is

$$.577 = 1 - (.25 \times .46) - \frac{.38 \times .46}{1.08} - \frac{.37 \times .46}{1.08^2} .$$

The first component represents the purchase of the system. The second is the tax benefit[10] for the first year's depreciation. The next is the discounted tax benefit for the second year's depreciation. The last is the discounted tax benefit for the third year's depreciation.

B. "Salvage Value of System A (f_{12})"

The next factor of option 1 is the salvage value of system A at the end of the four-year period. Our estimate of the salvage value is 30% of the purchase price, or $221,000. In the model, the value of this factor is -$221,000 since it represents a positive cash flow.

7. Note that f_{pj} and f_{qk} are the same common independent variable in the two cost equations.

8. If we are considering more than two options, we need to treat common factors in a pairwise manner with respect to the least cost option.

9. We assume an 8% corporate cost of money.

10. This example is taken from a real problem that was modeled during 1986. The tax rates and depreciation schedules used are taken from the problem and do not reflect the new tax laws for 1987.

In present value terms, the amount of this factor is

$$.429 = \frac{.54}{1.08^3}.$$

Note it is assumed that System *A* is fully depreciated for tax purposes at the end of the study. Therefore we realize only 54% of the expected salvage value.

C. "Salvage Value of System *B* (f_{13})"

We estimate the sell market value for system *B* as $800,000. The model factor value is therefore -$800,000. In present value terms, the amount of this factor is .54. This assumes that the system has been fully depreciated for tax purposes.

D. "Maintenance Cost of System *A* (f_{14})"

After the warranty period, we estimate the yearly maintenance cost as $8,280. In present value terms, the amount of this factor is

$$2.60 = .023 + \frac{1}{1.08} + \frac{1}{1.08^2} + \frac{1}{1.08^3}.$$

Note that the first component, .023, takes into account the warranty period.

E. "Environmental Cost of System *A* (f_{15})"

The estimated yearly environmental (space, power, cooling) cost of operating system *A* is $15,500. In present value terms the amount of this factor is

$$3.58 = 1 + \frac{1}{1.08} + \frac{1}{1.08^2} + \frac{1}{1.08^3}.$$

7.2 Option 2 Factors and Amounts

A. "Software Costs (f_{21})"

In order to use a production system for testing, we have to run a special software package that costs $60,000 per year. If we exercise option 2, this software will no longer be required. Thus this factor has a value of -$60,000.

In present value terms, the amount of this factor is

$$3.58 = 1 + \frac{1}{1.08} + \frac{1}{1.08^2} + \frac{1}{1.08^3}.$$

B. "Salvage Value of System *B* (f_{22})"

The salvage value of system *B* at the end of the four-year period is estimated as $10,000. The corresponding factor value is -$10,000.

In present value terms, the amount of this factor is

$$.429 = \frac{.54}{1.08^3}.$$

C. "Recovered Production Capacity (f_{23})"

Option 2 includes moving test work from a production system. Since we can reuse this capacity to meet an expected increase in our customer's service demands, we can avoid the installation of an equivalent amount of additional capacity. We estimate our installation cost per millions of instructions per second (MIPS) as $96,300. Thus, the value of this factor is -$96,300. We will move 2.5 MIPS of test work from the production system, so that the amount of this factor is 2.5.

D. "Maintenance Cost of System *B* (f_{24})"

We estimate the yearly maintenance cost of system *B* as $48,200. In present value terms, the amount of this factor is

$$3.58 = 1 + \frac{1}{1.08} + \frac{1}{1.08^2} + \frac{1}{1.08^3}.$$

E. "Environmental Cost of System *B* (f_{25})"

The estimated yearly environmental (space, power, cooling) cost of system *B* is $54,200. In present value terms the amount of this factor is

$$3.58 = 1 + \frac{1}{1.08} + \frac{1}{1.08^2} + \frac{1}{1.08^3}.$$

7.3 Example Linear Programming Formulation

The linear programming that models this decision problem is[11]

minimize

$$.735X_{11} - .221X_{12} - .800X_{13}$$

$$+.00828X_{14} + .0155X_{15} - .060X_{21}$$

$$-.010X_{22} - .0963X_{23} + .0482X_{24} + .054X_{25}$$

subject to

$$X_{11} - .577 \times D_1 = 0$$

$$X_{12} - .429 \times D_1 = 0$$

$$X_{13} - .54 \times D_1 = 0$$

$$X_{14} - 2.6 \times D_1 = 0$$

$$X_{15} - 3.58 \times D_1 = 0$$

11. The factors in the program are written in units of $1,000,000.

$$X_{21} - 3.58 \times D_2 = 0$$

$$X_{22} - .429 \times D_2 = 0$$

$$X_{23} - 2.5 \times D_2 = 0$$

$$X_{24} - 3.58 \times D_2 = 0$$

$$X_{25} - 3.58 \times D_2 = 0$$

$$D_1 + D_2 = 1$$

$$D_1 \geq 0$$

$$D_2 \geq 0 .$$

The solution to this model is

Decision Variable Values	
D_1	0
D_2	1

Stable Factor Ranges		
	Lower	Upper
f_{11}	.617	∞
f_{12}	-.38	∞
f_{13}	-.926	∞
f_{14}	-.0177	∞
f_{15}	-.0035	∞
f_{21}	−∞	-.041
f_{22}	−∞	.149
f_{23}	−∞	-.0693
f_{24}	−∞	.0672
f_{25}	−∞	.073

7.4 Some Observations About the Solution

- Option 2 is the least cost option.
- "f_{11}"

  When the decision is executed, if the purchase price of system *A* is 16% ($118,000) less than our estimate, then our decision is incorrect. This suggests that we should spend some effort to verify our $735,000 estimate.

- "f_{13}"

  When the decision is executed, if the salvage value of system *B* is 16% ($126,000) more than our estimate, then our decision is incorrect. Again, this suggests that we should spend some effort to verify our $800,000 number.

- "f_{21}"

  If we do not save at least $41,000 per year on software costs if option 2 is selected, then our decision is incorrect. We should verify that we will be able to terminate the required software on the production system.

- "f_{23}"

  We need to save at least $69,300 per MIPS by moving test work to system *B* in option 2. If the work cannot move or if our cost to install new MIPS falls below $69,300, then our decision is incorrect. This estimate probably needs additional verification.

- The other estimates in our analysis probably have sufficient accuracy for this decision problem.

8. Conclusion

A simple linear programming model for evaluating the financial aspects of the alternatives in an equipment acquisition decision has been developed. The model can be implemented on an AT&T PC6300 with an interactive linear programming system such as LINDO. The major benefit of using the model is the sensitivity analysis of the basic financial estimates that results from the solution of the linear programming problem. The model is easy to use and helps develop insight into the financial assumptions of the decision problem.

References

[1] R. Brealey and S. Myers, *Principles of Corporate Finance*, McGraw-Hill, Inc., 1981.

[2] G.D. Eppen and F. J. Gould, *Quantitative Concepts for Management*, Prentice-Hall, Inc., 1979.

[3] D. G. Newnan, *Engineering Economic Analysis*, Engineering Press, Inc., 1983.

[4] M. S. Bazaraa and J. J. Jarvis, *Linear Programming and Network Flows*, John Wiley & Sons, Inc., 1977.

[5] L. Schrage, *Linear Programming Models with LINDO*, The Scientific Press, 1981.

[6] L. Schrage, *Linear, Integer, and Quadratic Programming with LINDO - User's Manual*, The Scientific Press, 1986.

A comparative study of System/370 CPU's

Gilbert Houtekamer and Rob van Hoboken

Delft University of Technology,
Faculty of Electrical Engineering,
Mekelweg 4,
2600 GA Delft,
The Netherlands.

Guus Bonnes[1]

IBM IBS-SC,
Boerhaavelaan 11,
2713 HA Zoetermeer,
The Netherlands.

ABSTRACT

The MIPS rate of a processor is no longer a valid measure of system performance, because overall system performance depends on many factors. However, instruction timing measurements are still a powerful method to compare processors with the same instruction set. Instruction timing results can also provide interesting insight in the processor's internal operation. In this study the MIPSMON software monitor has been used to time most problem mode instructions on three processors using the IBM System/370 architecture. The results reveal some peculiarities of the processors involved. Instruction mixes, collected with the PPTRAC trace monitor, are used to estimate the MIPS rate in real programs: the instruction frequency and instruction timing data are combined to provide a weighted average. Finally, the potential of a cache analysis program based on the PPTRAC traces is explored.

1. Introduction

The performance of a computer system depends on many factors, among which the speed of the central processor, the size and access time of main and cache memory, the I/O configuration and the operating system. Early systems were characterized with the processor speed only, the power being expressed in terms of millions of instructions executed per second. This measure is called the MIPS-rate. In the current mainframe systems, however, the I/O subsystem and the operating system level have a profound effect on the system performance. Also, the MIPS-rate depends on the workload being executed on the processor.

For these reasons, manufacturers no longer specify a MIPS rate for their machines. Instead, the processor power is expressed as "so many times Model XX". IBM uses the "Relative Processor Power" (RPP), which indicates how fast a processor is, compared to the 370 Model 158-1. Incidentally, everybody knows that the 370 Model 158-1 is a 1 MIPS machine.

Measuring the instruction execution time for individual instructions does not provide a reliable measure for system performance, but it is a powerful method to evaluate the performance of the central processor component (CPU). Since the CPU speed is directly influenced by the cache and main memory speed and organization, the effects of these components will show up in the results as well.

The speed of a processor depends on many factors. [Hughes 1983] and [Bonnes 1983] provide an overview of these. The most important ones in this context are:

- The execution time for each instruction, as a function of the operand length(s).
- The instruction mix of the programs running on the central processor.
- The cache organization, access time and hit ratio, and the main memory organization and access time.

In this paper a method will be described for measuring the execution time of individual instructions using the MIPSMON software monitor, which has been developed at the Delft University [Bonnes 1983]. The measurement method eliminates most disturbances from cache and main memory. It will be shown that many characteristics of the processor hardware can be derived from the timing results.

In addition to the timing data on individual instructions, we have determined the instruction frequency for several applications, with a program called Problem Program Tracer (PPTRAC) [Paans 1986]. PPTRAC collects address and instruction reference streams, which are used to acquire statistics on instruction usage. The combination of the MIPSMON timing results and the instruction stream provides the relative CPU load of each instruction for a particular program mix running on a particular CPU.

The MIPS rate of a program depends not only on the instruction mix, but also on the address and instruction reference pattern. The pipelining in the processor, the cache implementation, the pattern of storage references and the translation look aside buffer all influence the processor MIPS rate. We will present a method for obtaining statistics on cache usage by a program based on the instruction reference streams. Many problems remain, however, and it is not possible to quantify the cache performance completely from the measurement data.

2. Description of the processors

All measurements have been performed on three implementations of the IBM System/370 architecture: the IBM 3033N, the IBM 3083B and the Siemens 7890D. The hardware implementation of these processors is quite different. Table 1 summarizes some of the characteristics of the processor hardware. These numbers are obtained from the documentation supplied by the manufacturers [IBM 1982a][IBM 1982b][IBM 1983][Siemens 1982], supplemented with estimates from our measurements[2].

1. Although the author is currently employed by IBM, the opinions expressed together with the recommendations made are strictly his own and have not been subjected to any formal IBM review.

2. main memory contains global buffer, access time depends on hit ratio

TABLE 1. Summary of the processor characteristics.

characteristic	processor			
	IBM 3033N	IBM 3083B	Siemens 7890D	
Machine cycle time	25/32	26	appr. 30	ns
cache access time	appr. 60	52	appr. 60	ns
cache size	16	32	16	Kbyte
cache block size	64	128	64	byte
main memory access time	342	312		ns
main memory size	16	16	16	Mbyte

The machine cycle time and the cache access time can be easily derived from the MIPSMON results, as will be shows below. In fact, the Siemens timing data were obtained from the measurements. The IBM 3033N and the Siemens 7890D use a store-through cache memory, while the IBM 3083B uses a flagged-swap cache memory. In a store-through cache memory, each write operation from the processor is immediately propagated to the main memory, blocking the cache memory for some time. In a flagged-swap cache, writes are propagated to the main memory only when the cache block contents are removed because the block is required for more recent data. The advantage of a flagged-swap cache is that it reduces the traffic between CPU cache and main memory significantly. A disadvantage is the more complex logic needed to implement the cache.

The Siemens 7890D uses a "global buffer" on the main memory, to reduce the main memory access time. In fact, it is an additional cache memory, be it somewhat slower and larger. In a multiprocessor 7890 system, each CPU would have its own cache ("local buffer"), but still use the same global buffer.

3. Measuring the Instruction Execution Time

The Instruction Execution Time (IET) is the time required to execute a single instruction. It is not possible to measure this time directly, except possibly with a hardware monitor. Even then it will be difficult to follow an instruction through the phases of instruction fetch, decode and execution as it flows through the processor pipeline.

The MIPSMON software monitor uses a different approach: the instruction to be measured is executed many times, and the total time required is measured using the CPU timer. The MIPSMON monitor consists of a driver program, which collects the timing information, and a separate module for each instruction to be measured. Each instruction mix module consists of 100 instructions of the type to be measured (and a subroutine return instruction). The driver calls the module N times, and measures the total time. Since the total time, and the number of times the instructions are executed are known, the IET can be computed. The measurement method is depicted in more detail in Figure 1.

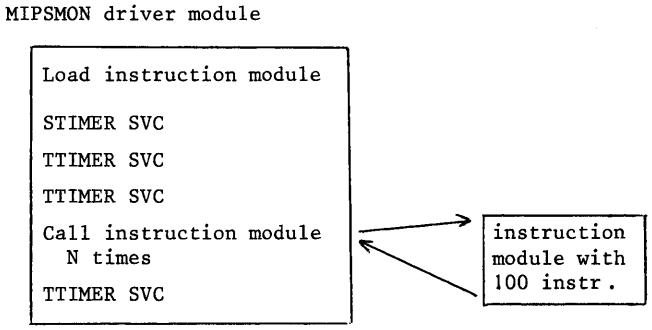


Figure 1. Overview of MIPSMON operation.

The MIPSMON driver consists of the command interpretation modules and the reporting modules, the code to load and execute the instruction mix and the timer facilities. The instruction mix module is loaded separately for each measurement.

The CPU time used is found by means of the timer services provided by the operating system. The MIPSMON driver first initializes the timer (STIMER), then it determines the timer overhead (first two TTIMER calls), and finally times the instruction mix (last TTIMER call). The MIPSMON driver will determine the average instruction execution time from the timer values and the number of calls to the module. It is assumed that the module contains 100 instructions.

The IETs found only depend on the processor speed and the cache access time, since only a tight loop is executed: all data referenced will be available in the cache memory after a single run through the loop. For most instructions, the main memory access time will have no influence on the IET obtained with MIPSMON. This is of course not a typical situation for a real program; therefor the results will thus be optimistic in that sense, but for this application it is more important that they are well defined and reproducible. MIPSMON jobs have been run at widely varying load, and the results always matched within 1%.

4. Instruction Timing Results

The IET has been determined for most of the problem mode instructions. In general, the instruction execution time depends on a large number of different factors, divided into the "history" of the instruction on the one side, and the instruction and its operands on the other side. In the interpretation of the MIPSMON results, we will only consider the instruction type and the operand length as variables. All other effects are neglected.

4.1 RR- and RX instructions

The fastest instructions on a System/370 processor are the Register-Register (RR) operations, which use registers only. Register-Indexed (RX) operations use a register and storage operand, which may require some additional effort from the processor. Both RR and RX operations may deal with integers (32 bit signed) and reals (64 bit floating point). Some other types are defined as well, but these will not be discussed here. Table 2 shows the timing results for load (L), add (A) and divide (D) instructions for the data and instruction types mentioned.

TABLE 2. Execution time for some RR and RX instructions (all results in nanoseconds).

instruction type		processor 3033N	3083B	7890D	
LR	RR, integer	60	54	60	ns
AR	RR, integer	60	54	60	ns
DR	RR, integer	1170	945	773	ns
L	RX, integer	88	80	60	ns
A	RX, integer	88	105	60	ns
D	RX, integer	1116	970	730	ns
LDR	RR, real	60	80	60	ns
ADR	RR, real	217	220	88	ns
DDR	RR, real	1750	1130	705	ns
LD	RX, real	88	132	60	ns
AD	RX, real	221	202	90	ns
DD	RX, real	1750	1156	710	ns

These results should be compared with the the CPU cycle time and cache access time from Table 1. The RR-type integer load and add instructions, with a 16 bit opcode, execute in two cycles. For the 3033 25+32 ns, for the 3083 2*26 ns and for the 7890 2*30 ns. This is also approximately the cache access time. The corresponding RX instructions, with a 32 bit opcode and a storage operand, require additional effort to read the operand. For the 3033 the execution time is now 88 ns (3 cycles) for both the load and add operations. The 3083 requires 80 ns (3 cycles) for the load, and 105 ns (4 cycles) for the store. The reason can only be that the 3083 requires an extra cycle to store the result from the add operation, after the memory operand has been fetched. The 7890 CPU is quite clever, it processes both RX-type operations at the same speed as their RR-type equivalents. Although not presented here, the results for the logical, subtract and compare operations are analogous for all CPUs.

Instructions for multiplication and division require considerably more CPU time. The limitation here will be the power of the arithmetic unit of the processor: a division requires a lot of cycles or a large number of circuits. For the integer division, the 3033 is the slowest machine (approximately 40 cycles), and the 7890 the fastest (approximately 26 cycles).

The floating point operations again show the power of the 7890 processor: the RR and RX load execute in two cycles, the adds in three. The 132 ns access time for the 3083 LD instruction is remarkable: why does the processor need 5 cycles for a simple load operation? It seems there is bottleneck in one of the internal busses of the processor, which slows down a fetch from memory.

The cache design is reflected most directly by the store operations. For a store-through cache, each store requires a main memory update, while a flagged-swap cache stores in the cache only. The data is propagated to main memory only when the cache block is needed for other data. Table 3 shows the timing results for the load and store operations.

TABLE 3. IET for load and store operations in nanoseconds.

instruction	processor 3033N	3083B	7890D	
Load (RX, integer)	88	80	60	ns
Store (RX, integer)	283	105	175	ns

The 3083 easily outperforms the other CPUs for store operations due its flagged swap cache. The 7890 is faster than the 3033, probably due to its global buffer. It is interesting to note that even for the 3033 the average IET of a store instruction is less than the specified main memory access time shown in Table 1 (283 vs 342 ns).

4.2 Jump Instructions

Jump and conditional jump instructions are important for the MIPS rate, since they occur frequently in all programs. Table 4 shows the IET timing results for some jump instructions.

TABLE 4. IET for various jump instructions in nanoseconds.

instruction	processor 3033N	3083B	7890D	
BALR (RR jump & save)	200	130	80	ns
BCTR (RR, jump not taken)	170	100	90	ns
BCTR (RR, jump taken)	170	130	130	ns
BCR 0 (no operation)	60	80	60	ns
BCR 15 (unconditional jump)	200	130	80	ns

The Branch And Link Register (BALR) instruction saves the next instruction address in the first register, and jumps to the address specified in the second register. The Branch on CounT Register (BCTR) instruction decrements and tests a register value, and branches if the registers contents are not equal zero. The Branch (Conditional) Register (BCR) instruction is a conditional jump instruction. The operand specifies the processor status bits to be used in the test (0 and 15 are special cases).

The 7890 is again the faster machine: unconditional or failing jumps take 3 cycles, and the conditional BCTR instruction takes 4 cycles if a jump a required. The 3083 is similar, in that a conditional branch instruction is faster if no branching is needed. It is remarkable that the BCTR executes as fast as the BCR 15 on the IBM machines, despite the register update required. The Siemens machine requires two additional cycles for the register update. The IBM machines will probably prepare the jump in parallel with the decrement and compare, while the Siemens serializes these events.

It is interesting to note that the 7890 processor documentation [Siemens 1982] states that 5 cycles are required for a successful jump, while the measurements suggest a value of 3. The BCTR instruction indeed takes four or five cycles. The difference might be caused by the MIPSMON measurement method, where all jumps skip only a few instructions and the destination instruction has already been processed in the pipeline.

4.3 Decimal Instructions

Instructions to process packed decimal numbers are relatively slow, because the format is not easily handled in a binary computer; to achieve a reasonable performance, a significant amount of dedicated hardware is required. The IETs for the decimal instructions differ significantly for the CPUs studied, as is shown in Table 5.

4.4 Block Move and Compare Instructions

The MVCL and CLCL instructions move and compare a block of data respectively. These instructions use 4 registers, to specify source and destination addresses and lengths. Table 6 presents the timing results for some block sizes, from 1 byte to 8 Kbyte.

TABLE 5. IET for some arithmetic instructions in nanoseconds.

		processor			
	instruction	3033N	3083B	7890D	
A	4 byte integer	88	105	60	ns
AD	8 byte real	221	202	90	ns
AP	4 byte decimal	890	459	381	ns
(add)	8 byte decimal	1111	560	380	ns
	16 byte decimal	1719	763	425	ns
ZAP	4 byte decimal	779	434	497	ns
(move)	8 byte decimal	1001	535	497	ns
	16 byte decimal	1499	763	599	ns
M	4 byte integer	281	382	162	ns
MD	8 byte real	520	405	132	ns
MP	4 byte decimal	2988	1738	1480	ns
(mult.)	8 byte decimal	5209	1471	1480	ns
	16 byte decimal	19031	8563	3565	ns
D	4 byte integer	1116	970	730	ns
DD	8 byte real	1750	1150	710	ns
DP	4 byte decimal	4793	7265	1467	ns
(divide)	8 byte decimal	16285	10908	2837	ns
	16 byte decimal	31415	33463	4312	ns

In this table it is remarkable that the 7890 processes 4 and 8 byte operands at the same speed (except for the divide, which requires a double width temporary value). The 7890 ALU must be able to process 8 byte quantities, where the 3033 and the 3083 are limited to 4 byte values. These results agree with those found for the RR and RX instructions, where the 7890 processed real numbers (8 byte) at the same speed as integers (4 byte), while the 3033 and 3083 required additional cycles.

The MP and DP instructions are the only instructions whose execution time is not linear in the operand length. This is a direct consequence of the multiplication and division algorithms used, which are quadratic in the number of digits processed, unless extensive hardware support is available.

The 7890 is equally fast for compare and move operations. An 8196 byte operation takes about 4000 cycles, or 2 cycles for each fetch or store operation on a 8 byte unit. Since the cache access width is about 8 byte, this will be the maximum rate supported by the cache.

In this time, not only the copy/compare operation has to be performed, but also the register update for the base address and counters. The CLCL and MVCL are interruptable and restart from the previous results, not from the beginning of the operation, so the registers must be kept up to date.

The 3083 requires roughly 6000 cycles for an 8196 byte copy, or 3 cycles for each fetch or store operation on a 8 byte unit. Probably this is due to limitations in the CPU hardware for the register update.

TABLE 6. IET for the compare long (CLCL) and move long (MVCL) instruction, as a function of the operand length. All timing results are in nanoseconds.

		processor			
	instruction	3033N	3083B	7890D	
CLCL	1 byte	2181	695	530	ns
	4 byte	2181	696	530	ns
	16 byte	2298	935	781	ns
	64 byte	3346	1834	1516	ns
	256 byte	7540	5322	4459	ns
	1024 byte	26009	20718	16247	ns
	8196 byte	216112	165065	129170	ns
MVCL	1 byte	2689	1919	1195	ns
	4 byte	2838	1919	1195	ns
	16 byte	2883	2105	1348	ns
	64 byte	3598	2905	2023	ns
	256 byte	6439	6095	4401	ns
	1024 byte	20145	22067	13945	ns
	8196 byte	131157	177347	131800	ns

TABLE 7. Instruction frequency top-10 reported by PPTRAC for 5 application programs (see text for details).

progr.	COBOL001		SMFPOST		APPLIC01		APPLIC02		SMFCOUNT	
#instr	1,000,000		2,000,000		183,078		2,779,581		284,901	
instr	opcode	%	opcode	%	opcode	%	opcode	%	opcode	%
1	BC	21.3	BC	21.4	BC	23.4	BC	18.6	BC	19.7
2	ZAP	20.3	L	8.8	L	10.3	L	11.0	L	10.0
3	CP	15.1	LA	6.6	TM	9.1	ST	6.1	TM	8.0
4	MVC	5.8	ST	6.4	LA	7.0	LA	6.1	LA	7.4
5	L	5.8	BCR	4.7	ST	4.7	MVC	6.0	MVC	6.0
6	SP	4.2	CLI	4.5	LR	3.9	CP	4.8	SR	4.8
7	DP	3.1	MVC	4.4	CLC	3.4	LR	4.3	IC	4.8
8	MP	2.9	MVI	3.3	MVC	2.9	SR	3.8	ST	3.8
9	XC	2.6	TM	3.2	MVI	2.7	AR	3.6	LR	3.5
10	LA	2.5	IC	3.1	SR	2.7	BCR	2.9	AH	2.5
# svc's	40	0.0	979	0.04	1523	0.8	185	0.0	287	0.1

The 3033 is "slow" for the compare operation, about 8000 cycles for an 8000 byte compare. Since the move operation is fast, the difference will probably be caused by limitations of the ALU in performing the compare operation. Another possibility would be that the processor is optimized for the move operation.

5. Collecting Instruction Traces

In the previous sections it was already mentioned that it is necessary to know which instructions occur how often (on the average) to determine the MIPS rate on a system. Unfortunately, it is impossible to collect this kind of statistics in a real system, on a real time basis. We have chosen an approach where a few programs, which are believed to be representative for the production workload on the system, are traced instruction by instruction. The traced instructions and their operands are then saved in a file for future analysis.

In System/370 computers it is not possible to trace a program on instruction basis with standard facilities: no hard- or software single step support is available to the user. At the Delft University, the software monitor PPTRAC (Problem Program Tracer) has been developed to cope with this problem [Paans 1986]. PPTRAC traces a program by executing it instruction by instruction: it determines which instruction has to be executed, loads the registers, executes the instruction and saves the registers. The instruction executed, its address and operands are saved in an output file. PPTRAC traps a few SVCs, changing the operands before the SVC is invoked. The SVC code itself is not traced.

During the trace process, an output dataset is created with the instruction addresses, opcodes and parameters. This data can be used to produce statistics on instruction frequency, to drive a cache simulator (based on the addresses), or it can be combined with the timing results from MIPSMON to provide a weighted MIPS-rate for the traced program.

The PPTRAC program has been used to trace 5 application programs. For each program, the top-10 instructions by frequency are shown in Table 7.

The APPLIC01 and APPLIC02 programs are database applications, written in COBOL. SMFPOST and SMFCOUNT are assembler programs to analyze SMF records, and COBOL001 is a COBOL program for SMF reporting.

It is remarkable that the conditional branch instruction (BC) is the most frequent instruction in all programs, at about 20%. Other frequent instructions are load (L, about 10%), store (ST), load address (LA), move character (MVC) and test (TM) each at roughly 5%. The COBOL001 program is a case in itself: this program uses packed decimal instructions extensively to handle decimal numbers. This program will be discussed in detail in a next section.

6. Computing the Real MIPS Rate

While the instruction frequency is interesting measure, it gives no indication of the processor usage of each instruction. A frequently used instruction may execute very fast, while another infrequently occurring, may be very slow, making its total CPU demand larger. Also, the execution time of an individual instruction depends very much on its operands: a compare over 8 bytes executes much faster than one over 8000 bytes.

TABLE 8. Processor utilization per instruction for the SMFPOST assembler program on the three CPUs.

no	IBM 3033N CPU			IBM 3083B CPU			Siemens 7890D CPU		
	opc	freq %	CPU %	opc	freq %	CPU %	opc	freq %	CPU %
1	BC	21.41	14.56	BC	21.41	16.71	BC	21.41	12.91
2	MVC	4.38	11.28	MVC	4.38	11.65	MVC	4.38	12.49
3	ST	6.42	8.23	ZAP	1.70	5.43	ST	6.42	9.29
4	ZAP	1.70	6.36	LM	1.60	5.10	ZAP	1.70	7.14
5	CLI	4.58	5.79	L	8.83	4.92	CLI	4.58	5.57
6	CLC	2.45	5.37	CLC	2.45	4.77	LM	1.60	4.93
7	STM	1.12	5.31	ST	6.42	4.69	L	8.83	4.38
8	MVI	3.26	4.13	STM	1.12	4.54	TR	0.01	3.70
9	AP	0.97	3.90	CLI	4.58	4.21	STM	1.12	3.63
10	BCR	4.65	3.89	BCR	4.65	4.03	LA	6.60	3.28
	MIPS		4.529	MIPS		6.956	MIPS		8.271

To cope with this problem, the relative CPU utilization should be reported instead of the instruction frequency. This number can be obtained from the instruction trace collected by PPTRAC, which contains all instructions and their operands, combined with the timing data collected by MIPSMON, being the execution time for all instructions as a function of the operand length. A separate report is now needed for each processor. Table 8 shows the top-10 based on CPU utilization for the SMFPOST assembler program from Table 7.

The ranking in these tables is quite different from Table 7. The branch instruction is still important, but the utilization is only 12-16% percent, as opposed to a 21% occurrence frequency. The store (ST) instruction ends up higher for the 3033 and 7890, and lower for the 3083 due to its flagged-swap cache operation. Expensive instructions like the ZAP, LM and STM, which are each quite infrequent (less than 2%), end up with utilizations between 3 and 7 percent. The translate (TR) is a case in point: an occurence frequency of 0.01%, and a CPU load of 3.7% on the 7890 CPU.

The bottom line in the table shows the MIPS rate as computed for these programs. These MIPS rates are more or less the "typical" values to be obtained for an application program on these processors.

In this paper only results for user-programs were reported. The full set of MIPSMON measurement results include some privileged instructions; also, PPTRAC has been used to trace some operating system functions in an MVS SE2 system. However, insufficient data was collected to predict the performance of operating system code in general. Data obtained so far indicates that instruction mixes of supervisor routines also consist mainly of the type of instructions found in user programs. Priviliged instructions are hardly used.

There are also fundamental problems involved in measuring privileged instructions: it does not always make sense to perform MIPSMON IET measurements on privileged instructions. Does it help you to know how much time it takes to execute a mix of 100 PURGE TLB instructions? (By the way, it takes 550 microseconds on a 3081-D [Paans 1986]).

7. A Case Study: Decimal and Binary Arithmetic

The COBOL001 program in Table 7 showes a very large fraction of decimal instructions, which are slow on System/370 machines. Table 9 shows the MIPS rate for this program, and the relative load for each instruction. The results are striking: the MIPS rate is down to 1.4 for the 3033 and 3 for the 7890.

The decimal instructions are very expensive to execute, which is reflected in the results in this table. The branch instruction (which is the most frequent one), is down to the 6th, 5th and 7th place for the 3033, 3083 and 7890 respectively. The COBOL compiler generates large numbers of decimal instructions like ZAP, AP, CP, MP and DP if the packed format is specified for variables. To investigate the influence of the format specification on the code produced, some of the more frequently used variables were changed to a binary format. A new trace shows a quite different

TABLE 9. Relative processor utilization and instruction frequency for the COBOL001 program.

no	IBM 3033N CPU			IBM 3083B CPU			Siemens 7890D CPU		
	opc	freq %	CPU %	opc	freq %	CPU %	opc	freq %	CPU %
1	DP	3.08	25.17	DP	3.08	42.63	ZAP	20.39	30.96
2	ZAP	20.39	23.59	ZAP	20.39	17.38	CP	15.07	16.98
3	CP	15.07	16.83	CP	15.07	12.49	MP	2.89	12.74
4	MP	2.89	11.12	MP	2.89	7.85	DP	3.08	11.98
5	SP	4.16	4.91	BC	21.32	4.12	SP	4.16	4.71
6	BC	21.32	3.82	SP	4.16	3.57	PACK	1.09	4.44
7	MVC	5.83	2.80	XC	2.63	2.71	BC	21.32	4.44
8	XC	2.63	2.67	MVC	5.83	2.04	MVC	5.83	2.89
9	AP	1.78	1.99	AP	1.78	1.48	XC	2.63	2.48
10	PACK	1.09	1.80	PACK	1.09	1.23	AP	1.78	2.01
	MIPS		1.393	MIPS		1.849	MIPS		2.983

TABLE 10. Relative processor utilization and instruction frequency for the improved COBOL001 program.

no	IBM 3033N CPU			IBM 3083B CPU			Siemens 7890D CPU		
	opc	freq %	CPU %	opc	freq %	CPU %	opc	freq %	CPU %
1	ZAP	9.07	16.57	DP	0.94	23.73	ZAP	9.07	19.20
2	CP	7.75	14.12	ZAP	9.07	13.25	PACK	2.23	12.75
3	DP	0.94	13.72	CP	7.75	11.31	CP	7.75	12.26
4	BC	19.65	6.64	BC	19.65	7.17	BC	19.65	5.96
5	PACK	2.23	6.13	PACK	2.23	4.50	DP	0.94	5.34
6	MP	0.80	5.19	MVC	5.30	3.95	MP	0.80	4.96
7	MVC	5.30	4.79	MP	0.80	3.83	CVB	1.99	4.76
8	CVD	2.96	3.86	CVD	2.96	3.74	CVD	2.96	4.73
9	AP	1.49	2.86	XC	1.43	2.62	MVC	5.30	4.46
10	XC	1.43	2.41	CVB	1.99	2.52	ST	3.37	2.48
	MIPS		2.328	MIPS		3.305	MIPS		4.203

program: from the PPTRAC instruction trace it appeared that the total number of instructions is reduced by a factor 2.2; and the MIPS rate increases by more than 50%. The total performance improvement is more than 400%. The top-10 results are given in Table 10. The decimal instructions are still important, and conversion instructions now show up (convert to binary (CVB) and convert to decimal (CVD)).

It is sometimes suggested that conversion between data types should not be used, since expensive conversion instructions are required to convert the different types into each other [IBM 1976]. From the MIPSMON results it appears, however, that even the simplest decimal instruction like AP takes as much time as a convert instruction. The conversion overhead will thus usually be low compared to the savings obtained by the use of binary arithmetic.

8. Scope of this work

In the analysis as described above, only the average execution time of instructions and their frequency of occurrence were considered. We did not take into account the instruction sequencing, we did not model the cache and translation look aside buffer, we did not incorporate task switching nor operating system behavior.

The MIPS rate achieved in a real system depends strongly on the pipelining effectiveness and on the percentage of translation look aside buffer and cache hits. These quantities, in turn, depend on the storage reference pattern of the programs running, and on context switching.

The effectiveness of the prefetching and pipelining hardware in the CPU critically depends on the instruction sequence; an unexpected "far" branch makes the preloaded information useless. Using instruction traces collected with PPTRAC, it would be possible to predict how often such branches occur. In a more general sense, one could adjust the execution time of an instruction, based on the previous one(s). The problem with these "high order" approximations will be in measuring of suitable timing adjustment values.

The performance of the cache memory depends on the locality of reference; the cache will buffer the information most recently accessed. When a new area of storage is being referenced, as occurs when a new task gains control of the processor, a copy of this storage will not be available in the cache. The task experiences a cache miss for each new address (or group of addresses) not previously referenced. Each cache miss requires a reference to main storage, which is usually much slower than a cache reference. During the execution of the task, the cache will be filled for this task, replacing data from previous tasks. The longer a task runs, the more data of previous tasks will be unavailable in the cache.

For the best overall cache performance, the average path length (the number of instructions executed without interruption) should be long. When a task gains control of the processor it may find storage areas from a previous execution still in the cache. This is of course dependent on the size of the cache and on how long ago this task was last active. In an MVS environment, the latter information could be obtained from a GTF trace [Bonnes 1984][IBM 1985]. A study on task locality is reported by [Kobayashi 1986].

With an appropriate approach it seems possible to combine the MIPSMON, PPTRAC and task-switch information into a single model of CPU behavior. Using the same approach, workload components like a swap-in could be described in terms of task-switches and instruction reference patterns.

9. Conclusions

The analysis method as presented provides a powerful method to compare the speed of processors in the IBM System/370 range. The approach is based on instruction timing measurements, in combination with instruction traces. The method can also be used to evaluate the quality and efficiency of code generated by compilers. An example has been discussed where the use of binary instead of decimal arithmetic yielded a performance improvement of a factor 4.

For analyzing the system performance the method must be extended to incorporate the cache and translation look aside buffer effectiveness, and the performance impact of task-switching. The major problem with this approach is that it seems almost impossible to collect trace data for operating system components in anything but a test situation.

10. References

[Bonnes 1983] A.H.J. Bonnes, **Instruction execution time measurements in MVS systems**, Computer Performance, Vol 4., No 3., September 1983, pp 167-175.

[Bonnes 1984] A.H.J. Bonnes **Determination of GTF performance overhead**, ECOMA-12 proceedings, 1984

[McHugh 1983] M.J. McHugh, **MIPS revisited**, Journal of Capacity Management, Vol 1., No 4., 1983, pp 372-381.

[IBM 1976] **OS PL/I checkout and optimizing Compilers: Language Reference Manual** IBM order no GC 33-0009

[IBM 1982a] **IBM 3083 Functional Characteristics**, IBM order no GA 22-7083, 1982

[IBM 1982b] **IBM 3033 Functional Characteristics**, IBM order no GA 22-7060, 1982

[IBM 1983] **IBM System/370 System Summary: Processors**, IBM order no GA 22-7001, 1983

[IBM 1985] **MVS/XA System Programming Library: Service Aids**, IBM order no GC 28-1159, 1985

[Kobayashi 1986] M. Kobayashi, **An Empirical study of Task Switching Locality in MVS**, IEEE Transactions on Computers, Vol C-35, No 8, August 1986.

[Paans 1986] R. Paans, **A close look at MVS Systems: Mechanisms, Performance and Security**, North-Holland Publishing Company, 1986, ISBN 0-44-70008-0

[Siemens 1982] **Siemens System 7.890, Summary Description**, Siemens order no U1000-J-Z62-1-7600, July 1982.

Tightly-Coupled Effect Processor Contention Overhead

Jerry P. Place
Computer Science
University of Missouri - Kansas City
5100 Rockhill Road
Kansas City, MO 64110
(816) 276-2359

Abstract. Tightly coupled parallel systems are increasingly prevalent. IBM's 3090-600E will have six tightly-coupled processors running under MVS. While these configurations of tightly-coupled processors have advantages such as straight-forward resource sharing and medium-granularity parallelism, there are some drawbacks. The drawbacks are called the *tightly-coupled effect*. The tightly-coupled effect states that a workload executed on a multiprocessor configuration can consume more processor overhead than the same workload executed on a uniprocessor. This paper will present one major aspect of the tightly coupled effect - processor contention for the dispatching queue - and will indicate the degree of the effect across different levels of total system utilization.

Key Words. data structure contention, shared-memory multiprocessing, tightly coupled multiprocessing, dispatching, process scheduling.

1. Introduction.

Tightly-coupled configurations of computing machines are widely available today from several vendors. It has been recognized that tightly-coupled configurations of computing machines offer substantial benefits such as data sharing, device sharing and medium granularity parallelism. Parallel configurations are also seen as a possible alternative to bigger and more powerful single processor systems, i.e. instead of installing a bigger, more powerful uniprocessor, install a tightly-coupled configuration and allow these machines to work together to solve problems.

Control in tightly-coupled systems fits into one of three categories *master/slave, separate supervisor,* and *floating control*[1]. The *master/slave* mode assigns one of the processors in the configuration as the master. The operating system kernel routines always execute on the master. The master is responsible for dispatching work to the slave processors. The *separate supervisor* method of control maintains a copy of the operating system kernel for each processor in the system. Other system entities such as tables and common routines must be accessible through shared memory or a shared file system. The *floating control* method maintains one copy of the operating system and each processor executes that one copy. Thus, each processor is responsible for satisfying its own requests for service. Floating control systems are the most difficult to design and implement. The single copy of the operating system must be reentrant and, since key system data structures must be shared by all processors in the configuration, constructs to ensure determinism must be embedded in the operating system.

In order to ensure determinism in a system, only one processor at a time will be allowed to access a common data structure when several processors require read/write access to the structure. While there are many resources within a tightly-coupled operating system that fall into this category, this paper will focus on the system dispatching queue. If $processor_a$ cannot gain access to this structure because $processor_b$ has exclusive access, $processor_a$ must wait - *spinning on the lock.* that guards the queue. Figure 1 depicts a configuration of processors about a shared memory. Each process in the system is divided into its component tasks. All tasks competing for resources in the configuration are represented by task control blocks in the dispatching queue. The task control blocks indicate the status of the task - either dispatchable or not, and the dispatching priority of the task. Since processors share all system resources, the processors are considered equivalent, thus any task may execute on any processor - regardless of the tasks that are executing on other processors. It is possible that several tasks from the same process will be executing in parallel on different processors in the system. As a task executes, events occur that cause it to be blocked - perhaps an I/O request, or a page fault. When an executing task is blocked, its processor is context switched, i.e. another task is *dispatched* to that processor.

The dispatching mechanism is enacted by each processor executing the shared copy of the dispatching process. In essence, floating control in this specific example means that each processor must find its own job to execute next - independently of all the other processors. A processor finds its next unit of work by searching the *system dispatching queue* - generally implemented as a linked list of task control blocks. Since each processor will update one or more task control blocks on the dispatching queue, i.e. mark the task control block as *blocked* when the task is blocked for I/O or a page fault, the dispatching queue must be a serially reusable resource in order to ensure determinism in the system[2]. Thus, in order to access the dispatching queue, a processor must exclusively own the *dispatching lock* which guards the queue, i.e. only one processor at a time may use the dispatching queue.

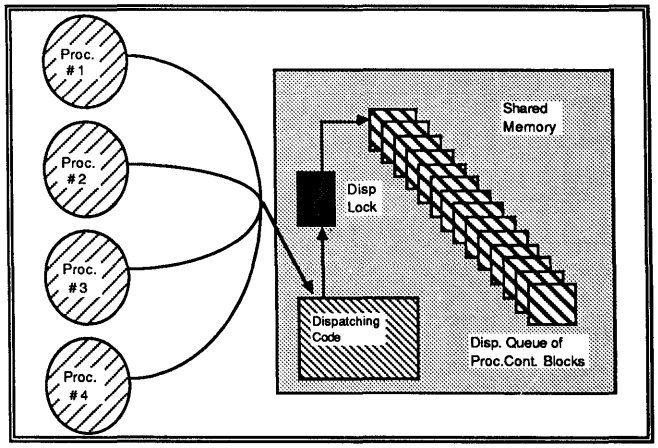


Figure 1 - Tightly-Coupled Configuration

2. System Data Structure Contention.

The shared *READY* queue is a point of contention among processors in the configuration[3]. This contention is part of measured CPU utilization in a tightly-coupled system and constitutes significant overhead per processor which is sometimes referred to as the *tightly-coupled efect*[4,5]. Figure 2 presents a simple queueing model that reflects the contention for the dispatching queue. The assumptions of the model are:

- Ts_1 is the dispatching queue service time, $Ts_1 = 13\mu s$.
- Ts_2 is the average time that each process will execute before a context switch,

  $Ts_2 = 500\mu s$.

• Ts_3 is the time each processor can expect to spend idle, Ts_3 is determined from the input.

• All processors are homogeneous.

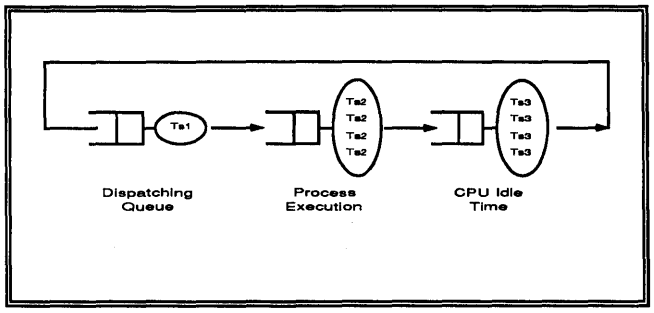


Figure 2 - Processor Scheduling Queueing Model

Only one processor at a time may access the dispatching queue. It takes each processor Ts_1 seconds to find a task to execute. When a processor finds its next job, it executes an average of Ts_2 seconds then the processor is idle for Ts_3 seconds. Assumptions for this simple model are based upon an average of 200 task control blocks on the dispatching queue and a memory cycle time of 100ns.

Figure 3 presents the results of executing the model at average processor utilization levels of 20%, 40%, 60% and 80%. The model then was executed for tightly-coupled configurations of one through sixteen processors - at each average utilization level. The chart provides a measure of the impact of increasing contention for the dispatching queue as the number of processors in the configuration increases. The percentage amounts in the chart were computed as $\frac{W_1}{W_1 + W_2}$ where W_1, W_2, and W_3 are computed wait times at each node in the model. The percentage amounts in the chart may be interpreted as the percentage of execution time (CPU utilization overhead) that could be attributed to the dispatching function. It is emphasized that the percentage is on a per-processor basis, thus 3.56% contention overhead on each processor in a configuration of sixteen processors means that about 3.56% * 16 = 0.5696 or just over 1/2 of a processor is consumed by dispatching queue contention across the parallel configuration.

Average dispatching contention overhead ranged from a low of 1.3% of CPU execution time per processor to 3.56% per processor (see figure 3). As expected, the lowest queue contention overhead occurred at low average processor utilization and the highest contention overhead was recorded with a configuration of sixteen processors each with an average utilization of 80%.

3. Conclusion

This study is important for several reasons. Contention overhead is an important factor that must be considered when deciding whether to run a system loosely-coupled or tightly-coupled. Also, contention overhead could be an important consideration when reviewing a possible tightly-coupled configuration that will replace an existing uniprocessor.

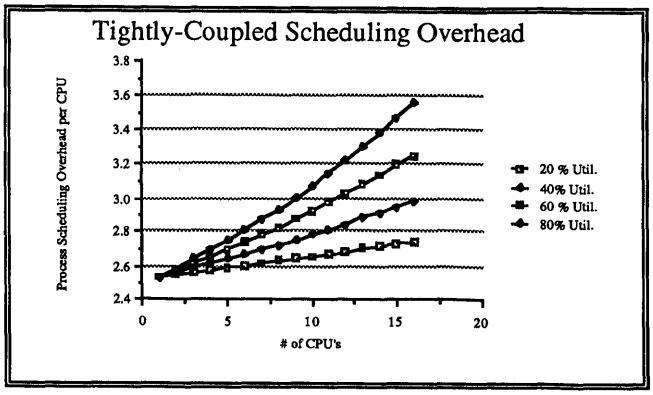


Figure 3 - Processor Scheduling Overhead

The actual overhead percentages obtained by this study are not particularly important. The average task execution time between context switches is dependent on the nature of the system workload. The most important point made by this study is the relationship between dispatching queue contention overhead and the number of processors in a tightly-coupled configuration and the impact that average processor utilization can have on the contention overhead.

Bibliography

[1] - K. Hwang and F. A. Briggs,*Computer Architecture and Parallel Processing* , McGraw-Hill, New York, NY, 1984, 846 p.

[2] - E. G. Coffman and P. J. Denning, *Operating Systems Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1973, 331 p.

[3] - J. P. Place, The system data structure contention problem and a novel software solution for shared-memory, floating-control parallel systems, **Proc 1987 Nat Comp Conf, 54**, June 1987, pp. 109-117

[4] - K. L. Radecki, Introduction to Processor Evaluation, **Tech Bul GC66-0232-00**, IBM Washington Systems Center, Gaithersburg, MD

[5] - D.A. Smucker, MVSE: Tightly Coupled Systems, **Proc. SHARE '85**, Feb. 1985, pp. 585-592

FORECASTING CPU UTILIZATION IN AN IMS ENVIRONMENT: A CASE STUDY USING MONTE CARLO TECHNIQUES

W.T. Gray, M. Cheng
Canadian Pacific

ABSTRACT

Canadian Pacific runs three large 3090 class processors dedicated to IMS. Over one hundred separate production applications execute well over 2 million transactions per day. To forecast future resource requirements, the capacity planning group has evolved a sophisticated forecasting technique.

In a quarterly process, the project managers for each IMS application provide a forecast of the average number of transactions which they expect to see executed each day. Aided by historical actuals and their previous estimates, they return a forecast with it's upper and lower bounds. This forecast and sampled system data is input to a Monte-Carlo simulation. The output from the simulation is a distribution of projected CPU utilizations. From this we can make a statement like: we expect a machine to be 70% busy by the end of the fourth quarter but there is a 20% probability it will be 85% busy.

In comparing actual results with forecasts from previous periods, we see that the process has correctly bound the results. The procedure also has the virtue of wide management acceptance.

OVERVIEW

In the first section of this paper, we describe how transaction forecasts are gathered. The second section discusses the calculation of CPU utilization from these forecasts. The next section describes random variables and the application of the Monte Carlo technique to this problem. In the next section, we discuss extentions of the technique to a more complex environment. Following this, we describe how to perform sensitivity and "what if?" analysis. Finally, we discuss the general applicability of the technique, its strengths and weaknesses.

FORECASTING

There are many different ways to forecast demand. In our case it is possible to get a good forecast from the user community out six quarters. This is a fortunate not only from the point of view of accuracy, but also of accountability.

Whenever you go to a user for a forecast, there is a problem. You want CPU seconds and they have invoices processed. You can use their metric and translate it to yours by some multivariate technique. You can train the user to think about CPU time. This can work if the user is billed in units of CPU time. We make a compromise and forecast in terms of transactions. The user community understand transactions. They are a simple concept and the user can relate them to work units within their individual environment. They are an ideal compromise for us in that we can track transactions and the CPU time they consume. They also have the advantage that they do not change when the processor or the operating system changes. Non-technical managers accept them for their simplicity and because they can add them together over multiple CPUs and multiple datacenters. This is useful in depicting overall load changes, especially when we move applications between processors or sites.

From a capacity planning point of view, transactions are quite a different matter. For one thing, they're all different. Each consumes different resources, at different times of the day, on different machines, in different regions. The problem for the capacity planner is translating a transaction forecast into a forecast of resource utilization.

There are two ways to group transactions: by some manner of resource usage, or by application. From a performance point of view, the best way to deal with transactions is by some resource consumption category (eg. small, medium, large). For performance purposes, it is immaterial which user submitted the transaction. For Capacity Planning, the size of the transactions is immaterial. The size of the application is the critical element. We do not classify transactions within application. We could request that the user forecast the number of small and large transactions within their application, but we feel that this is unnecessary. We only require that the mix remain somewhat constant, and this is usually the case.

There are many sources of variability in the transaction volume for an individual application. Not only does the transaction volume vary from day to day, but the project manager's forecast is only a guess. To understand the variability, we ask that they supply a minimum guess, a maximum guess, and a "most likely" guess. To help guide the project managers in this task, we provide the last six quarters of actuals and their last forecast. The format of all this data is minimum-mean-maximum. At each iteration of the process we ask that they look at the previous forecast and change it if they see fit. At the same time we ask them to forecast out another quarter such that the forecast always extends six quarters.

Figure 1 is an example of the type of form which we send to the project managers. The last three columns represent the last forecast

supplied to us. The project manager has not seen the 87Q2 actuals before. We have added the 87Q4 blank line. The project manager is to examine the forecast in light of the actuals, modify it if he or she sees fit and estimate one more quarter.

```
Application Name    : AAA
Project Manager     : BBB
Appl ID Code        : CC
Lookalike           : N/A
Percent Prime Shift: XX
System Run On       : DD

      Min   Mean   Max   Min   Mean   Max
86Q1  123    419   730
86Q2  107    552   912
86Q3  201    631   886   100    500   800
86Q4   99    478   934   100    500   800
87Q1   97    459   956   100    500   800
87Q2  225    682  1087   100    500   800
87Q3    .      .     .   100    500   800
87Q4    .      .     .   100    500   800
88Q1    .      .     .   100    500   800
88Q2    .      .     .   100    500   800
88Q3    .      .     .   100    500   800
88Q4    .      .     .     .      .     .
```

Figure 1. Outgoing forecast form.

Figure 2 shows how we may get this form returned. Note that the project manager has revised the estimate upwards.

```
Application Name    : AAA
Project Manager     : BBB
Appl ID Code        : CC
Lookalike           : N/A
Percent Prime Shift: XX
System Run On       : DD

      Min   Mean   Max   Min   Mean   Max
86Q1  123    419   730
86Q2  107    552   912
86Q3  201    631   886   100    500   800
86Q4   99    478   934   100    500   800
87Q1   97    459   956   100    500   800
87Q2  225    682  1087   100    500   800
87Q3    .      .     .   200    750  1200
87Q4    .      .     .   200    750  1200
88Q1    .      .     .   200    750  1200
88Q2    .      .     .   200    750  1200
88Q3    .      .     .   300    900  1600
88Q4    .      .     .   300    900  1600
```

Figure 2. Returned forecast form.

The corporate office automation system is the medium for mailing and returning these forms. We maintain the forecast information on a database, accessible to both the electronic mail system and our SAS based programs. The forecast collection portion of the system runs almost completely under program control such that there is no lengthy key entry step.

PROCESSOR MEASUREMENTS

When a machine is running on average, 80% busy across prime shift and if prime shift were 10 hours long, then BUSY time is 8 hours and IDLE time is 2 hours. When we refer to forecasting percent utilization we are talking about forecasting the number of hours of BUSY time. Figure 3 below shows schematically the allocation of all CPU time. This figure represents a pure IMS environment. In other words, there is no TSO or batch.

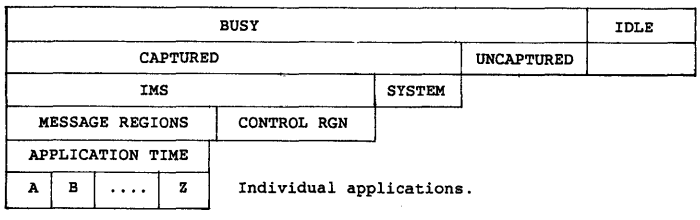


Figure 3. Schematic of time recording on "pure IMS" machine.

Within BUSY time, there is time which the operating system cannot allocate to any specific task; rather, it is work done for the entire system. Consistent with convention, we refer to this as UNCAPTURED time and the ratio of CAPTURED time to BUSY time as the SYSTEM CAPTURE RATIO.

$$\text{SYSTEM CAPTURE RATIO} = \text{CAPTURED}/\text{BUSY} \quad (1)$$

Since RMF records BUSY time, we can always calculate UNCAPTURED time by subtracting CAPTURED time from BUSY time. The best way to calculate the total CAPTURED time is to sum all the CPU time from each performance group.

We can divide CAPTURED time among the tasks which use it. In our example, we decompose this time into IMS and SYSTEM. We define SYSTEM as the operating system and started tasks (MVS/JES, VTAM, ACF, etc.). Within the IMS portion, we include the message regions and the control region. Within the message regions, we find the APPLICATION time. SMF records this time for each transaction in these regions (SRB plus TCB). This is normally the only time which we bill back to the user because it is only at this level that CPU time can be attributed back to an individual transaction. In our site, APPLICATION time and billable time are synonymous. It is instructive to note that the sum of the APPLICATION time which SMF records is slightly less than that recorded by RMF in the message region performance groups.

In a pure IMS environment we call the ratio of APPLICATION time to BUSY time the APPLICATION CAPTURE RATIO. Any fluctuation in this ratio is caused by fluctuations in transaction mix. In a mature system, this capture ratio is stable.

$$\text{APPLICATION CAPTURE RATIO} = \text{APPLICATION}/\text{BUSY} \quad (2)$$

Another important metric is APPLICATION CPU time per transaction. We calculate this for all transactions within an application and not across the entire system.

$$\text{AVERAGE TRANSACTION TIME} = \text{APPLICATION}/\text{TRANSACTION COUNT} \quad (3)$$

We examine not only the stability of the capture ratio but also the stability of the AVERAGE TRANSACTION TIME. In mature applications, the transaction mix is relatively constant and so are these two metrics. There are applications in which the mix is seasonal. We can allow for seasonality if we recognize it. We do not have any applications in which the seasonal change in AVERAGE TRANSACTION TIME is significant. The most variable applications are those which are growing rapidly. These are usually still in the implementation phase, and are picking up new offices and/or new functions. We recommend that capacity planners carefully monitor growth applications, not only to track transaction count but also to see if the transaction mix is as expected. If the transaction mix changes during the implementation of a new application, then the resources which the application ultimately consumes may be considerably different than that forecast.

Later, we will describe how we handle new applications.

For stable applications, we make the assumption that recent behavior of AVERAGE TRANSACTION TIME will be the same out to the forecast horizon. Multiplying this metric by the forecast transaction volume will yield the CPU time for that application. The subscript i indicates an individual application.

$$\text{APPLICATION}_i = \text{TRANSACTION COUNT}_i * \text{AVERAGE TRANSACTION TIME}_i \quad (4)$$

If we repeat calculation for each application, and sum the results, then we arrive at total APPLICATION time.

$$APPLICATION=SUM(TRANSACTION\ COUNT_i*AVERAGE\ TRANSACTION\ TIME_i) \quad (5)$$

Multiplying the total APPLICATION time by the APPLICATION CAPTURE RATIO yields CPU BUSY time.

$$BUSY=APPLICATION/APPLICATION\ CAPTURE\ RATIO \quad (6)$$

This was the number that we started out to calculate. Have we solved the problem? Yes and no. We have just forecast that a CPU will be X% busy in the 4th quarter of next year. How wrong is the forecast? Could the number be X+20%? If X were 83%, then we might have an interesting 4th quarter. We have not solved the problem until we can bound the forecast. If we were forecasting with some time series technique, then statistically it is a simple matter to put bounds on the forecast. We simply tell SAS that we want the 95% confidence limits. When we gather forecasts from users we are not dealing with a time series, we a dealing with a hundred different guesses.

RANDOM VARIABLES

The variables in the above discussion are random variables. Each has an associated frequency distribution. We talk in terms of averages, yet we never see an average day. If we monitor the transaction volume for an application over a period of time, we see a pattern similar to that shown in figure 4. This is a 100 day sample.

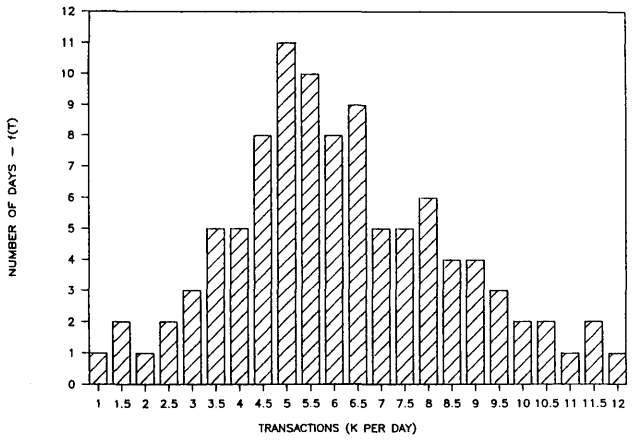


Figure 4. Frequency distribution of transaction arrivals

We observe there were 2 days which saw 2K transactions executed, 11 days at 5K, and 1 day at 12K. Although the average number of transactions run per day was 6K, there was quite a spread. Figure 5 shows the data in an accumulated frequency distribution.

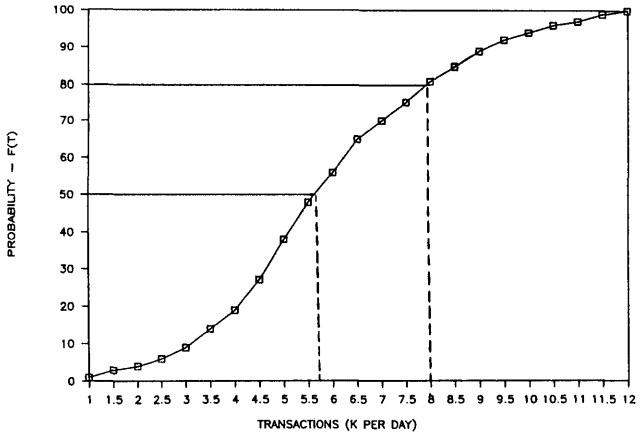


Figure 5. Accumulated frequency distribution

What is most instructive about this curve is that although the average transaction count is 6K per day, there is an 20% probability that the transaction count will exceed 8K. In other words, 1 day in 5 is a heavy day. If there are one hundred different applications running, many of them will have the same heavy day.

If we were to monitor the other variables, we would notice a similar pattern. CPU per transaction, AVERAGE TRANSACTION TIME, fluctuates the same way. We must expect this fluctuation to continue into the future. We track all the key metrics like CPU time per transaction and the various capture ratios over a period time. Quarterly, we calculate their mean and standard deviation. We treat all these measured variables as Gaussian or normal random variables. This is not an assumption. Since we are dealing with statistics calculated from a large number of samples, the Central Limit Theorem assures us that these variables will tend to be normally distributed.

In contrast to the measured variables, we must deal with forecasted numbers from individuals. To account for variability in the forecast, we insist that all forecasts be provided as: lowest possible, most probable, highest possible. As figure 6 demonstrates, a variable defined in this manner is called a triangular random variable. Figure 6 represents the forecast for the application shown in figures 4 and 5. The lowest guess is 3, the most probable 6.6, and the highest guess 12.

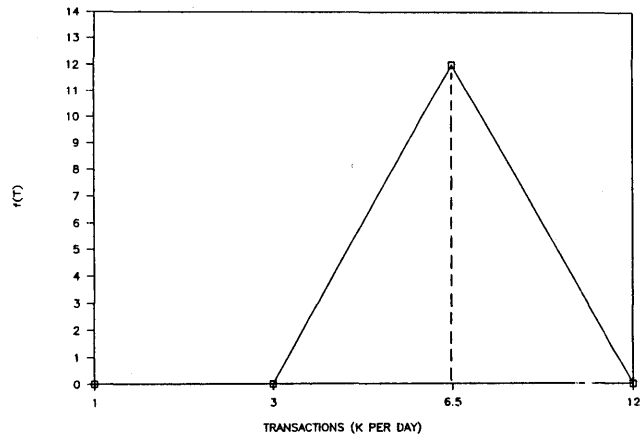


Figure 6. Example of Triangular Random Variable.

Our problem is one of performing the calculations in equations 2 through 6 with random var-

iables. To do this we use a technique called Monte Carlo simulation. The technique was developed by the operations research groups during WWII in England, when they actually threw dice to create random numbers. Later the dice were replaced by tables of random numbers. Now every computer and calculater can quickly and painlessly produce a random number.

Throwing a die extracts a random number between 1 and 6 drawn from a uniform distribution; all results are equally likely. In real world situations, the uniform distribution is quite rare and we require a method of extracting a number from other distributions. In our case, we need random numbers drawn from both normal and triangular distributions. Many different transformations exist to change a uniform random variable into other types. We can program these transformations ourselves, or write the simulator in a language, like SAS, which produces normal or triangular variables on request.

If we want to simulate any given day from the application above, we can ask for a triangular number with the parameters 3, 6.6, 12. If we repeated this calculation 200 times, then we would get a frequency distribution very similar to figure 6. We could simulate figure 4 by asking for 100 normal random variables with a mean of 6 and a standard deviation of 3.5. The essence of a Monte Carlo simulation lies in repeating a calculation many times, inserting a random number drawn from the given frequency distribution in place of the random variable.

Assume we have only two applications, one which has a min, mean, and max TRANSACTION COUNT forecast of (600,1000,1700) and the other of (900,2300,6000). The CPU per transaction, the AVERAGE TRANSACTION TIME, for the first is 20 ms with a standard deviation of 12 and for the second, 35 ms with a standard deviation of 19. Using equation 5, we calculate total APPLICATION TIME by summing the product of TRANSACTION COUNT and AVERAGE TRANSACTION TIME for the two applications. The * in the following equations depicts a random variable.

$$\text{APPLICATION}^* = \text{SUM}(\text{TRANSACTION COUNT}^*_i * \text{AVERAGE TRANSACTION TIME}^*_i) \quad (5a)$$

We get TRANSACTION COUNT for the first application from a triangular distribution with the parameters 600,1000,1700 and the AVERAGE TRANSACTION TIME from a normal distribution with mean 20 and standard deviation 19. We follow the same procedure for the second application. Figure 7 shows a sample run of 5 iterations.

Application A			Application B			
Trans Volume	CPU/ Trans	Appl. Time	Trans Volume	CPU/ Trans	Appl. Time	Total Time
1200	17	20400	1123	32	35936	56336
921	13	11973	2670	41	109470	121443
1403	24	33672	2354	29	68266	101938
753	29	21837	4200	37	155400	177237
1298	22	28566	3521	30	105630	134186
						118228 Average

Figure 7. Sample Monte Carlo Output

If we repeated this calculation 100 times, we would get an output distribution for total APPLICATION TIME similar to that shown in figures 4 and 5. In our shop, we iterate 200 times such that for each forecast period we end up with 200 random days within each period. By dividing APPLICATION TIME by the APPLICATION CAPTURE RATIO we get 200 values for CPU BUSY time. We reduce these 200 days by calculating the key statistical parameters, mean, standard deviation, 95th percentile, etc..

Assume that the threshold busyness for a dedicated IMS environment is 80%. In other words, we plan on upgrading a processor whenever it exceeds this average threshold. We can arrive at this threshold by experience or by modelling. The idea being that when the CPU is busier than 80% on the average then at peak periods response will deteriorate. Assume for a moment that we do not use the Monte Carlo technique and rather do a straight calculation using equations 2 through 5 and the most probable forecast value. Further assume that this calculation yields a forecast for average CPU utilization at 75%. Since this is well below the threshold, we inform our managers that they need not be consider an upgrade within the forecast horizon. If the forecast is low, then there will be some rude surprises.

Now lets do the same calculation using the Monte Carlo technique. We should get the same answer, 75%. In addition, however, we get a probability distribution which shows that, although the average is 75%, there is a 40% probability that the CPU utilization will exceed 80%, and a 10% probability that it will exceed 90%. We can only achieve this result, the probability distribution of processor utilization, by using a simulation. Capacity planners must give this critical information to management and they must use it as the basis of resource planning.

For applications which are still in the planning stage, it is not possible to measure any of the metrics which are essential to the forecast. In these cases, we have developed the concept of the look-a-like application. Developers inform us of an application which is similar to the one which they are working on. They give us their TRANSACTION COUNT forecasts and we measure the CPU time per transaction from the look-a-like. In a shop which has many applications and different transactions, finding a look-a-like is not a major problem.

EXTENSIONS

Let us now consider a more complex configuration. Suppose that we run some DB2 and some batch on a processors and that all the workloads are growing. Figure 8 below shows the CPU measurement diagram for this.

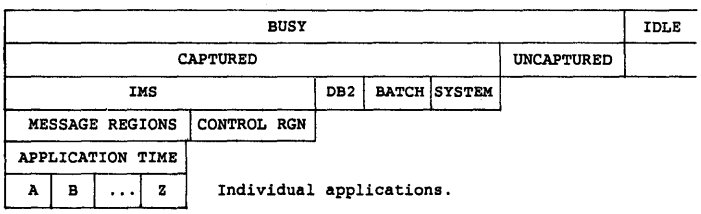


Figure 8. Schematic of time recording on mixed mode machine.

We calculate APPLICATION TIME exactly as equation 3. The rest is different. In equation 7 we calculate a ratio, the IMS RATIO, which is APPLICATION TIME divided by IMS TIME (the sum of the CPU time in all the IMS performance groups). We observe this ratio over time and, as before, measure the parameters of its distribution. In the simulation calculations, we work backwards and use this ratio to arrive at IMS TIME.

```
IMS RATIO = APPLICATION TIME / IMS TIME                    (7)
IMS TIME" = APPLICATION TIME" / IMS RATIO"                 (8)
```

We are now in the third layer from the top in figure 8. We will assume that techniques exist to forecast DB2, BATCH, and SYSTEM CPU time at this level. For example, one possible way would be to forecast user connect time and track CPU time per connect hour. We can then forecast DB2 time by:

```
DB2 TIME"=Forecast CONNECT time" * CPU time per connect hour"   (9)
```

We nest this calculation in a Monte Carlo program similar to the way we develop IMS time. Lets assume that the BATCH time and SYSTEM times are forecast by a trend line, and available to the program.

At this point we must calculate capture ratios. The UNCAPTURED time in figure 8 is not constant nor can we distribute it proportionately across the 4 subsystems. In this environment, the following calculation is NOT correct.

```
BUSY=(CAPTURED + UNCAPTURED)
    =CONSTANT * CAPTURED
    =CONSTANT * (IMS + DB2 + BATCH + SYSTEM)
```

If this were true, we would have a much simpler problem. (Note that CONSTANT is equal to 1/SYSTEM CAPTURE RATIO in equation 1.) Unfortunately, the system uses UNCAPTURED time in differing amounts for each of the four subsysems. In other words, each of the subsytems has a different capture ratio. The following calculation IS correct.

```
BUSY=(CAPTURED + UNCAPTURED)
    =CONSTANT * CAPTURED
    =BETA1*IMS + BETA2*DB2 + BETA3*BATCH + BETA4*SYSTEM        (10)
```

The values beginning with BETA are constants. Continuing the same convention, we call the reciprocal of the BETA, a capture ratio. The BETAs, as shown in equation 10 can only be calculated by multiple regression. Performing this procedure is relatively simple with SAS. Now that we have all the data required, we can forecast CPU utilization by putting equation 10 into the simulator. We treat the BETA variables as normal random variables. The regression procedure which calculates the BETA values for us also provides the standard deviations. In the regression, the standard error of the parameter estimates is the standard deviation of the capture ratios.

```
BUSY"=BETA1*IMS" + BETA2*DB2" + BETA3*BATCH" + BETA4*SYSTEM"   (10a)
```

SENSITIVITY AND "WHAT-IF" ANALYSIS

Let us assume a more complicated system has 5 different workloads: IMS, IDMS, DB2, BATCH, and SYSTEM (CICS and TSO are on another system!). We have a detailed Monte Carlo model of this system which uses equations similar to equations 7 through 10 above. In our analysis of the output of the 200 iterations of the simulation we can calculate how much the result varied for a 10% change in each of the input variables. This is a sensitivity analysis and it may look something like this:

```
a 10% change in "IMS application X"  generates a  4% change in "BUSY"
a 10% change in "IMS application Y"  generates a  8% change in "BUSY"
a 10% change in "BATCH"              generates a  2% change in "BUSY"
a 10% change in "IDMS application Z" generates a 10% change in "BUSY"
                        .                          .
                        .                          .
                        .                          .
a 10% change in "DB2 connect time"   generates a 22% change in "BUSY"
```

Our whole exercise was most sensitive to the variation in "DB2 connect time". This fact will become extremely valuable for management control.

"What If" we added another IMS application. We insert the forecast transaction volume and look-a-like application as before. Perhaps we would increase the standard deviation of the different variables to take into account our lack of certainty.

Because of the way we have structured this model, it is easy to modify parts of it and rerun. As such, "what if?" analysis is very easy. To "what if?" we increased the size of the CPU, we would simply multiply all the CPU factors by the constant which relates the size differences. To "what if?" we moved some applications to another system, we change the system identification field in the input data and the forecast is automatically inserted into that systems forecast.

An evolution to the system which we are currently working on is that of extending the forecasts out two years beyond the horizons that the developers are willing to forecast. We are designing this enhancement with time series and by modeling expert business analyst guesses. We are also currently adding IDMS to the forecasts and the simulations.

The technique is not without its problems. The shrinking application development window implies that we may miss an entire application. Our developers forecast only those applications which they are working on. It is possible for an application to be written and implemented within the forecast horizon and yet be unknown to all developers at forecast time. The inability of a developer to forecast applications which he is unaware of creates a flattening of the forecast curve out near the horizon. This flattening can be extremely misleading. From figure 9 we see that the forecast line can cross our planning boundary anywhere during 5 quarters. This effect can be minimized by keeping the forecast horizon the same length as the development window.

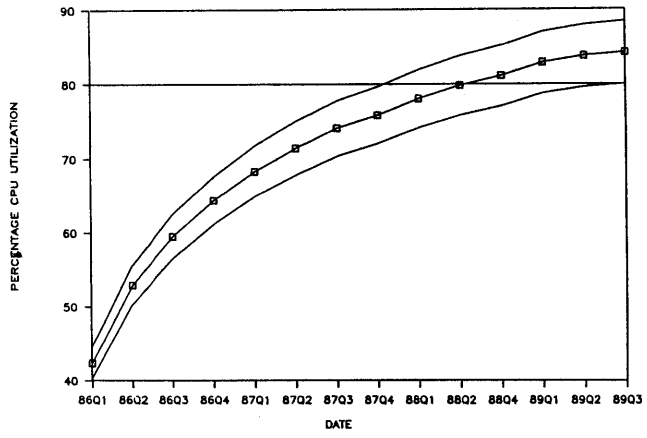


Figure 9. Sample forecast showing flattening effect.

CONCLUSION

We have been using this procedure for two years and we feel that it's greatest strength lies in it's ability to output probabilities of percent utilization. The increased amount of information which goes into the forecast makes possible the increased amount of information which it yields. The additional work can be automated and is worth the effort. The high credibility afforded to the results is due to the involvement of the users and developers and the amount of work they put into their input. Good forecasts are only produced with good data and good analysis. We recommend that other capacity planning groups adopt this technique for any forecast which is comprised of guesses from developers and users. It's flexibility and accuracy have brought it wide management acceptance.

Note: SAS is a trademark of SAS Institute Inc., North Carolina.

REFERENCES

H. Kobayashi (1978). "Modeling and Analysis: An Introduction to System Performance Evaluation Methodology". Addison-Wesley.

H.W. Merrill (1984). "Merrill's Expanded Guide to Computer Performance Evaluation Using the SAS System". SAS Institute Inc.

MODELLING OF A PRODUCTION IMS SYSTEM UNDER A MULTI-FUNCTION PROCESSING ARCHITECTURE

Dale Gilbert
Lomas & Nettleton Information Systems

ABSTRACT

IMS production systems at Lomas & Nettleton Information Systems (LNIS) utilize the facilities of the Future Systems Architecture (FSA). The FSA is an umbrella structure which provides access to terminal and data base functions for application programs without the programmer being cognizant of specific parameters or syntax. Under the FSA each IMS transaction may process many different types of requests based upon the "function code" submitted. Multiple instruction paths may exist within the same module making accurate transaction characterization difficult.

Analytic models of the FSA and production IMS systems are constructed using detailed analysis of trace and logging information. These models are used as a basis for forecasting future processing requirements for new applications under development.

INTRODUCTION

An architecture within IMS or other environmental software is used to centralize functions or operations common to other related applications in single area or set of programs. In addition such functions as uniformity of logon procedures across applications and security checking at a transaction level may be implemented within the architecture.

There are two general approaches to architecture design illustrated in Figure 1. A horizontal approach places the architecture components at the same level as the application programs. The programs are called directly from the environmental software and communicate with the architecture components and other applications through architecture calls. A vertical structure or umbrella approach places the architecture components between the environmental software and the application programs. The programs are called by means of program-to-program message switching within the architecture. This paper is concerned with the latter architecture currently in production at LNIS. A future use of this procedure will be to evaluate a horizontal implementation of the architecture.

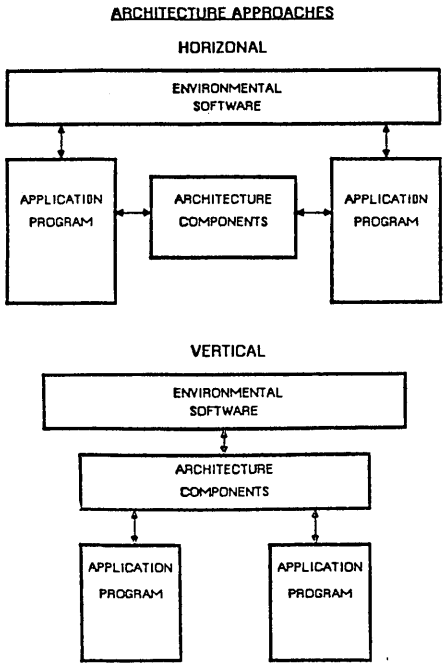


Figure 1. Alternative Approaches to IMS Architecture.

The vertical architecture has the following processing flow. When a terminal message is received by IMS it is queued against the Message Control Facility (MCF). The MCF program then analyzes the message based on flags/codes that exist in the architecture Scratch Pad Area (SPA) for the particular terminal and may select one of four options:

1) Pass the message to an application program. The appropriate driver associated with the targeted application program is selected based upon information contained within the current function code. The new function code information is assembled and a message switch is then made to the appropriate driver program with the incoming message appended.

2) Process the message as an MCF command. MCF calls the associated routine to process the request and returns an informational message to the terminal.

3) Process the message as a function code switch. MCF checks the access security of the user against the requested function code and creates a message switch to the appropriate driver/application program.

4) Process an unsolicited message. MCF interrupts the current conversation with the user and displays the message. Depressing 'Enter' causes MCF to continue the conversation.

A driver is a control program which selects the application to process a given message. The driver appears to IMS as an application program with its own Program Specification Block (PSB) and the corresponding Program Communication Blocks (PCBs). In turn a specified number of application programs are associated with each driver. These application programs are selected by function code and do not appear at the environmental software (IMS) level. The databases (DBDs) for each application must be defined at the driver level.

When a message is switched to a driver, the architecture SPA is again analyzed for path information and the corresponding application program called to service the message. Upon return, an interrogation of the architecture SPA determines whether 1) a second program within this driver, 2) an additional message switch, or 3) a response output to the user is required. When a message switch is required the cycle is repeated for the second driver. The output message ends the cycle until a new input message is received.

When a terminal message is received by IMS it is queued against a driver's transaction code. The driver then analyzes the Request Entry Line portion of the message to determine if a command or a function switch has been

entered. If the message is a command the associated routine is called to process the request and an informational message returned to the terminal. If the message is a function code switch then a message switch is made to the appropriate driver and the application program indicated by the function code.

Therefore, the Future Systems Architecture may be considered a multi-function processing architecture in which a single 'transaction' module may process many different types of requests based upon the function code submitted. Multiple instruction execution paths may be used within the same module.

OBJECTIVE

LNIS has and is developing a number of IMS applications planned to execute under the Future Systems Architecture described above. Given the projected magnitude of applications under development and expected business growth, a need exists to determine if service levels can be met and what are the hardware requirements to deliver the service levels.

The approach decided upon was to model in detail existing applications that execute under the architecture and from this effort develop procedures to aid in modelling the developmental applications.

SYSTEM DESCRIPTION

The host hardware environment consisted of an IBM 3090-400 processor and a combination of 3380's and 3380E's as primary I/O devices. The initial modelling effort took place with an IBM 3090-200 which was approaching its capacity in terms of the total workload. The difference in the processors provided additional model verification.

INSTRUMENTATION

The initial step was to identify the characteristics of each function code instruction path. A user executed a script based on the function codes belonging to the target application within the IMS test region. During the user session the IMS/VS DC Monitor was turned on to capture the DL/I calls. In addition the IMS log for the same time period was captured. The log included records from the IMS Measurement Facility (IMF). Each function code submitted during the session had the same driver, e.g., the transactions identified by the IMS log and IMF data were all the same name. However the sequence of execution given by the ITASK number within each message region of the records was sufficient to associate each function code with its resource consumption and its DL/I call pattern as shown in Figure 2. The DC Mon records are sorted by message region, ITASK number, and time stamp. The ITASK number is assigned by the IMS control region in order that IMS resources become available to processing programs and thus is the order of execution of the DL/I calls within the transaction.

The IMF data identifies the drivers (programs) called by the function code from within the architecture by the alternate PCB created by the program-to-program message switch. The alternate PCB name is the same as the driver name. The initial transaction and the subsequent driver invocation both have the same arrival time stamp. When the data is sorted by arrival time stamp, message region, and user identification the message switch transactions will be paired. The architecture transaction that generated the function code transaction sequence of Figure 2 is shown in Figure 3.

The DC Monitor data used in this analysis are the type 10 and type 60 records. The type 10 records are generated when a message scheduling event is started. These records identify the transaction and provide the initial time stamp and message region. The type 60 records are generated by DL/I calls and provide the bulk of the information. The type 60 records, when sorted by ITASK, reveal the DL/I call pattern of that transaction. The DL/I call is identified by the PCB name, the type of call, the segment name being retrieved, and the return status of the call. In addition, if any physical I/O's are made, the DBD name of the database and the time spent in IWAIT are recorded.

The paired transactions from the IMS log and IMF data can be interleaved with the DC Mon data by the transaction name, message region, and the successive time stamps of each source. The resulting data contain a complete picture of the architecture application interface. The architecture overhead consists of a transaction that is paired with each unique application transaction. The application can be modeled and evaluated separately from the architecture overhead. The overhead can then be added to give a complete performance picture.

The remaining information required for the model was the identification of the function code that the driver represented. Each driver is capable of multiple tasks according to a particular function code definition. Modelling at the driver level would not permit sufficient insight to generalize the model components and extend them to future applications. Some function codes within the drivers are peculiar to applications and must be identified.

Upon examination of the IMS log data it was discovered that the function code was carried in the type 01 and type 03 records. These records contain the screen transmission records in which the current date is followed by the function code the operator entered. Thus by using the time stamp of the log record to match with the IMF transaction records and the DC Monitor trace the transaction can be identified at the function code level. It is also true that this procedure can be extended to the production IMS environment in a strictly passive mode; no hooks are required to be placed in a production system. Using the same procedure within the production IMS system also would give the frequency of occurence of each function code, a necessary quantity to calibrate the model.

DATA COLLECTION

A peak period of transaction activity was selected for data collection. The sampling interval was limited to eight minutes due to the volume of data collected and the degredation upon system performance produced by DC Monitor. The eight minute interval produced 76,000 DL/I calls from about 1100 recorded transactions.

The type 60 DC Monitor records provided the detailed description of each DL/I call. The type 60 record contains the PCB name, DL/I function call, segment name, segment level, return code status, and any I/O wait due to a physical I/O. If no IWAIT trailer is present then no physical I/O took place. The number of IWAIT trailers in the record is then the number of physical I/O's per DL/I call. Additionally, when an IWAIT occurs the DBDNAME is carried in the trailer record along with the IWAIT time to complete the I/O.

When the type 60 record is combined with the type 10 record by ITASK number the PSB name is obtained. The PSB name is the same as the driver or transaction name. Combining again, this time with the IMF data by message region, transaction, and time stamp gives a complete transaction description down to the DL/I call pattern.

An additional benefit from looking at the detail type 60 records is the breakout of I/O to secondary indices. The IWAITS per DL/I call due to secondary index I/O is lost in the summarization process of the DC Monitor Call Summary Report. In the detail record the IWAIT is identified by the DBD name of the secondary index database

permitting the physical I/O's of the model to be directed against the correct database with the same PCB.

The collected data is now sufficient to build modules representing each of transactions/function codes occuring during the trace period. In summary the data consists of:

1) Driver and function code
2) Database being accessed
3) DL/I call sequence
4) DL/I call type
5) CPU time per function code use
6) I/O time per DL/I call
7) Physical I/O's per DL/I call
8) Secondary index use per DL/I call

The remaining data required to parameterize a DL/I call are the database type, organization, and placement. The task was simplified since all application database organizations under the FS architecture are VSAM. Dataset placement is an important consideration in IMS performance analysis and in its impact upon system performance. The data collection effort identified all active databases from their DBD name. The Database Administration function maintains a VSAM definition file in which each DBD name is associated with its cataloged dataset name, organization, block size, and its volume residency. Due to ocassional dataset movements for performance or other reasons, this definition file is read and a database catalog built as a module of the system description file.

MODEL CONSTRUCTION

Modules of each function code can be constructed. A particular function code can be represented by summarizing the monitor data by type and frequency of DL/I call. This will give an accurate representation of each function code but the summarization process loses the call pattern. The call pattern is useful for insights into program logic and to maintain a continuity with the design process - the end product.

Figure 2 shows a sequence of DL/I calls from a specific function code (DMI). When the monitor data is sorted by ITASK number for each function code and transaction with each message region, the DL/I call sequence of the original IMS transaction is obtained.

All transactions using the architecture begin their database call sequence with an I/O to the FSA spa to record the timestamp and other information the programmer sees fit to maintain. Following the call to the spa the sequence of calls belonging to the function code appear. The number of physical I/O's per transactions is the single most important component of response time. In the monitor data any time an I/O is performed the DBD name is recorded. The summarization process accummulates the I/O's for each database. In the example in Figure 2 the impact of using secondary indices to access data is recorded. The DL/I call identified by ITASK number 3F3AE8 obtains a loan number from the mortgage loan database MTGP001D. The next ITASK number (3F3AED) is repeated twice. The first line represents the I/O to get the required information from the Funds database. The next occurrence of the ITASK number contains the I/O required for the secondary indices. These I/O's are recorded by the DC Monitor to the same ITASK record. Three physical I/O's are required to obtain the necessary references to identify the desired record in the Funds database.

Another typical sequence of calls is shown in the next sequence beginning with ITASK 3F3B00. Calls are made to a database containing keys to another database. The initial call (a Get Unique) to the key database (TR2Y001I) requires two physical I/O's. An alternating sequence of DL/I calls then begins using the key information to identify the requested record in the master database (TR2P001D). After the initial GU call to the key database the I/O is performed using GN (Get Next) requiring, in most cases, no physical I/O (the record is already in the buffer). A physical I/O occurs with the GN when the record is not in the buffer and must be retrieved from DASD. In this sequence each call to the master database requires a GU and a corresponding physical I/O. The sequence ends when a status code of GE is returned indicating the root level has been reached and no additional information exists.

The remaining calls within this transaction interact with the database DLMPHSTD. An initial GU requiring one physical I/O is made to DLMPHSTD to obtain the level one segment DLMHMTG, a delinquent mortgage. A search is then made through the level two segments with GNP (Get Next within Parent) requiring no physical I/O until a level three segment is obtained (DLMQAGN). Following the segment DLMPQAGN call are two Get Unique's to the PMF tables. The PMF tables or Parameter Management Facility contain the format of the segment and enable the calling program to decode the segment into valid information. This technique makes the application less dependent upon specialized code but at a cost of one physical I/O per decoded segment. The status code of GE is returned whenever the segment level reached in the search increases, e.g., a GNP returns another parent rather than a child of the current parent.

Once the desired record is found additional information is collected from other databases and a screen is filled. At the conclusion of the transaction a replace (REPL) is made to the FSA spa and calls made to I/O PCB.

The transaction of Figure 2 requires 38 physical I/O's to complete the DL/I call sequence. At a nominal 3380 service time of 25 milliseconds per access the I/O component of the response time of this transaction should be 0.95 seconds.

One problem arising in looking at individual transaction call patterns is that many of the calls are database dependent. In the previous example the number of calls for a twin chain search depends upon the number of individual segments within a parent. If data from a number of such transactions are collected in the monitor interval an overall average transaction call pattern can be obtained.

Figure 4 shows the relative frequence of occurrence of function code use within driver DLMP100 using the IMS log type 03 records. The total hourly arrival rate is represented by the driver. The remaining function codes are the components of the driver, i.e., the sum of the arrival rates of the function code equals that of the driver. These values translate directly into model transaction arrival rates. The total CPU utilization of the IMS application is shown in Figure 5. Again the utilization of the driver is the sum of the utilizations of the function codes called by the driver.

Figure 6 shows the response times for these function codes from the IMF data when the individual transactions are identified from the IMS log data. The CPU consumption per function code use is computed from the data represented in Figure 4 and 5 and is shown in Figure 7.

Due to the volume of trace data a procedure was developed to create the simulated DL/I calls in a format necessary to represent each function code call as an input to an analytic model which described the internal IMS 1.3 pathlengths. The DL/I call patterns are generated using the processed DC Mon trace and IMS log data. The data were summarized by function code and frequency of occurrence of each function code. The modules developed in this manner could be referenced from more general workload descriptions within the model. Additionally, the performance of the driver modeled as a composite of function codes could be compared to standard reported measurements.

Figure 8 shows the modelled results of driver response broken into its components of CPU, I/O, and memory. Breaking the driver into its function codes gives the results in Figure 9. A more detailed look at response time as calculated from the model is shown in Figure 10. Each function code's residency time is computed at each of the system components.

A still more detailed output is shown in Figure 11 at the device level. The utilization of a particular device can be broken into database activity. This simplifies analysis of response time problems resulting from database contention within the driver and architecture applications.

The detailed output at the device level was useful during model calibration. The I/O devices were all 3380 D & E's with an estimated value for average seek time of 16 milliseconds. The response time component due to the CPU was within acceptable limits when comparing model output with monitor output. However, the I/O component was significantly higher for many of the devices. Using a tool designed to measure DASD performance provided actual seek times for each device within the model. The seek times ranged from 4 to 10 milliseconds for the production IMS packs which included both databases and program libraries.

The IMS production databases are subject to frequent tuning efforts to ensure optimum database performance (response time) through minimum arm movement. Including these estimates into the model produced response times within acceptable limits.

The resulting model can be evaluated against projected changes in both the architecture and existing applications. As demonstrated earlier the architecture transactions are broken out separately from the application and proposed changes can be implemented independently of or in conjunction with each other. Models of proposed applications similar to the application in this study can be constructed using the modules created from the detailed DL/I trace data. As shown in Figure 12 the model output reflects both the architecture and the application effects when proposed transaction growth is included.

The response times from various level of IMS activity were compared to the modelled results in Figure 13. During the period of model construction the processor was upgraded from an IBM 3090-200 to an IBM 3090-400. The response times from the model 400 activity show a very good agreement with reported values. The response data obtained from the Model 200 environment reflected the capacity limits that not only IMS was experiencing but also the remaining 80% of the workload. The IMS application and, as a result, the output reflects a dedicated processor running at a 20% busy level. To account for the elongated response times observed with the Model 200 a performance model of the complete system is required. The model of the IMS workload can be incorporated into a total workload description and a truer picture of service level performance obtained.

CONCLUSIONS

A procedure for evaluating the performance of existing IMS applications that conform to a multi-function processing architecture umbrella has been outlined. The procedure uses a number of generally available IMS performance monitor sources to produce modules representative of individual function codes that exist under the architecture and are not readily available from standard reporting procedures. The modules are developed both to model the architecture effects and to provide an interface with new application designers. Estimates of the performance and resource requirements of applications under development can be obtained by constructing analytical models using similar type modules with new database and system descriptions. The entire IMS production workload can then be evaluated using a higher level analytic model. The complete workload can be analyzed in terms of response times and throughput (transaction processing) in a resource contention environment by using a performance model with the modelled workload included. The end result should be a true picture of expected application performance in a specific processing configuration.

ITASK	PCBNM	FUNC	LEVEL	SEGMENT	STATUS	DDNAME	PHY_IO
3F3AE3	FSAPSPA	GU	2	FSASPA1		FSAPSPAD	1
3F3AE8	MTGP001	GU	1	MTGLOAN		MTGP001D	1
3F3AED	MTGPFDS	GU	1	MTGBCFM		MTGPFDSD	1
3F3AED	MTGPFDS	GU	1	MTGBCFM		MTGXFDS1	3
3F3B00	TR2Y001	GU	1	TR2AKEY		TR2Y001I	2
3F3B0F	TR2P001	GU	1	TR2INFO		TR2P001D	1
3F3B1B	TR2Y001	GN	1	TR2AKEY			0
3F3B1C	TR2P001	GU	1	TR2INFO		TR2P001D	1
3F3B1D	TR2Y001	GN	1	TR2AKEY			0
3F3B1E	TR2P001	GU	1	TR2INFO		TR2P001D	1
3F3CE9	TR2Y001	GN	1	TR2AKEY		TR2Y001I	2
3F3CEA	TR2P001	GU	1	TR2INFO		TR2P001D	1
3F3CF2	TR2Y001	GN	1	TR2AKEY			0
3F3CF3	TR2P001	GU	1	TR2INFO		TR2P001D	1
3F3CFB	TR2Y001	GN	1	TR2AKEY			0
3F3CFC	TR2P001	GU	1	TR2INFO		TR2P001D	1
3F3D0D	TR2Y001	GN	1	TR2AKEY			0
3F3D0E	TR2P001	GU	1	TR2INFO		TR2P001D	1
3F3D18	TR2Y001	GN	1	TR2AKEY			0
3F3D19	TR2P001	GU	1	TR2INFO		TR2P001D	2
3F3D36	TR2Y001	GN	1	TR2AKEY			0
3F3D37	TR2P001	GU	1	TR2INFO		TR2P001D	1
3F3D3E	TR2Y001	GN	0		GE		0
3F3D3F	DLMPHST	GU	1	DLMHMTG		DLMPHSTD	1
3F3D43	DLMPHST	GNP	2	DLMDDCK			0
3F3D45	DLMPHST	GNP	2	DLMDDCK			0
3F3D46	DLMPHST	GNP	2	DLMDDCK			0
3F3D48	DLMPHST	GNP	2	DLMDDCK			0
3F3D49	DLMPHST	GNP	2	DLMDDCK			0
3F3D4A	DLMPHST	GNP	2	DLMDDCK			0
3F3D4B	DLMPHST	GNP	2	DLMDDCK			0
3F3D4C	DLMPHST	GNP	2	DLMDDCK			0
3F3D4E	DLMPHST	GNP	2	DLMDDCK			0
3F3D4F	DLMPHST	GNP	2	DLMDDCK			0
3F3D50	DLMPHST	GNP	2	DLMDDCK			0
3F3D51	DLMPHST	GNP	2	DLMDDCK			0
3F3D52	DLMPHST	GNP	2	DLMCAHM			0
3F3D53	DLMPHST	GNP	3	DLMQAGN			0
3F3D54	DLMPHST	GU	3	DLMCAHM			0
3F3D55	DLMPHST	GNP	2	DLMCAHM	GE		0
3F3D57	DLMPHST	GNP	2	DLMCAHM	GE		0
3F3D58	FSAPDGD	GU	2	FSADFD			0
3F3D59	FSAPPDR	GU	1	FSAPDR		FSAPPDRD	1
3F3D5F	DLMPHST	GNP	2	DLMCAHM	GE		0
3F3D60	DLMPHST	GNP	2	DLMCAHM			0
3F3D61	DLMPHST	GNP	3	DLMQAGN			0
3F3D62	FSAPDGD	GU	2	FSADFD			0
3F3D66	FSAPPDR	GU	1	FSAPDR		FSAPPDRD	1
3F3D6E	DLMPHST	GNP	3	DLMQAGN			0
3F3D70	FSAPDGD	GU	2	FSADFD			0
3F3D72	FSAPPDR	GU	1	FSAPDR		FSAPPDRD	1
3F3D73	DLMPHST	GNP	3	DLMQAGN		DLMPHSTD	1
3F3D7A	DLMPHST	GNP	3	DLMQAGN			0
3F3D7C	DLMPHST	GNP	3	DLMQAGN			0
3F3D7E	DLMPHST	GNP	3	DLMQAGN			0
3F3D80	DLMPHST	GU	2	DLMCAHM			0
3F3D82	DLMPHST	GNP	2	DLMCAHM	GE		0
3F3D83	DLMPHST	GNP	3	DLMPPIP			0
3F3D86	DLMPHST	GNP	2	DLMCAHM	GE		0
3F3D88	FSAPDGD	GU	2	FSADFD			0
3F3D8A	FSAPPDR	GU	1	FSAPDR		FSAPPDRD	1
3F3D8E	DLMPHST	GNP	3	DLMQAGN			0
3F3D8F	DLMPHST	GNP	3	DLMQAGN			0
3F3D90	FSAPDGD	GU	2	FSADFD			0
3F3D91	FSAPPDR	GU	1	FSAPDR		FSAPPDRD	1
3F3D96	DLMPHST	GNP	2	DLMCAHM	GE		0
3F3D97	DLMPHST	GNP	2	DLMCAHM			0
3F3D99	DLMPHST	GNP	3	DLMQAGN			0
3F3D9A	FSAPDGD	GU	2	FSADFD			0
3F3D9B	FSAPPDR	GU	1	FSAPDR		FSAPPDRD	1
3F3D9E	DLMPHST	GNP	3	DLMLFSO			0
3F3DA0	DLMPHST	GNP	2	DLMCAHM	GE		0
3F3DA2	DLMPHST	GNP	3	DLMQAGN			0
3F3DA4	FSAPDGD	GU	2	FSADFD			0
3F3DA5	FSAPPDR	GU	1	FSAPDR			0
3F3DA6	DLMPHST	GNP	3	DLMLFSO			0
3F3DA7	DLMPHST	GNP	2	DLMCAHM	GE		0
3F3DA8	DLMPHST	GNP	2	DLMCAHM			0
3F3DA9	MGRP001	GU	1	MGRMTGR		MGRP001D	1
3F3DAC	MGRP001	GU	0		GE	MGRP001D	1
3F3DB0	PTYP001	GU	1	PTYLOAN		PTYP001D	1
3F3DBF	PTYP001	GNP	2	PTYPROP			0
3F3DC1	PTYP001	GNP	2	PTYPROP	GE		0
3F3DC3	FSAPDGD	GU	2	FSADFD		FSAPDGDP	1
3F3DD8	FSAPPDR	GU	1	FSAPDR		FSAPPDRP	1
3F3DD8	FSAPPDR	GU	1	FSAPDR		FSAYPDRI	1
3F3DF1	FSAPDGD	GU	2	FSADFD			0
3F3DF2	FSAPPDR	GU	1	FSAPDR		FSAPPDRP	1
3F3E00	MGRP001	GU	1	MGRMTGR			0
3F3E01	MTGP001	GU	2	MTGSTAT		MTGP001D	1
3F3E08	MTGP001	GU	2	MTGORG			0
3F3E09	MTGP001	GU	1	MTGLOAN	GE		0
3F3E0A	MTGP001	GU	1	MTGLOAN	GE		0
3F3E0C	MTGP001	GU	1	MTGLOAN	GE		0
3F3E0D	FSAPSPA	REPL	2	FSASPA1			2

Function DMI Call Patterns

SUBTYPE=ARCHITECTURE

ITASK	TRANSACT	MSGREGN	PCBNM	FUNC	LEVEL	SEGMENT	STATUS	DDNAME	PHYIO
3F3ACE	FSATMCF1	IMSFMR06	I/O PCB	GN					0
3F3AD0	FSATMCF1	IMSFMR06	FSAPSPA	GU	01	FSASPA		FSAPSPAD	1
3F3AD2	FSATMCF1	IMSFMR06	FSAPFCD	GU	01	FSAFCD		FSAPFCDD	2
3F3AD6	FSATMCF1	IMSFMR06	FSALTRM	GU	00		GE	FSAPTRMD	1
3F3AD9	FSATMCF1	IMSFMR06	FSAPSPA	REPL	01	FSASPA			1
3F3ADA	FSATMCF1	IMSFMR06	I/O PCB	CHNG					0
3F3ADE	FSATMCF1	IMSFMR06	I/O PCB	ISRT					0
3F3ADC	FSATMCF1	IMSFMR06	I/O PCB	GU				FSAPSPAD	1

Figure 3. Architecture DL/I Call Sequence

Function Code Use

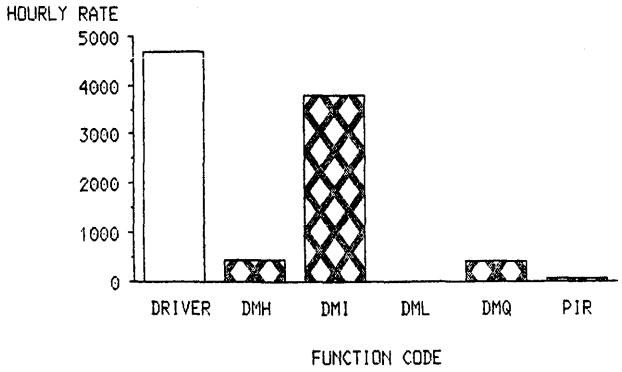


Figure 5. Function Code CPU Utilization (IBM 3090-400)

Function Code Response Time Breakout

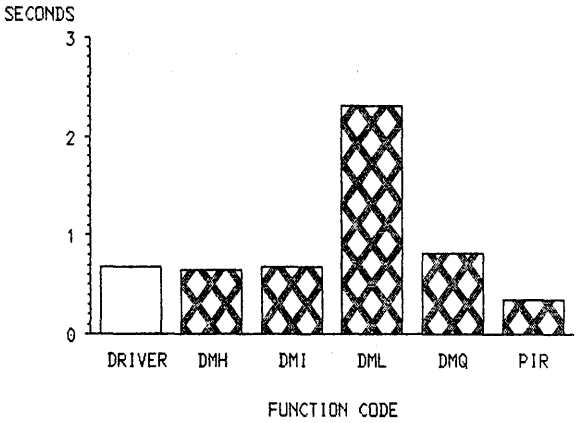


Figure 6. Measured Reponse Times Per Function Code

Function Code CPU Seconds

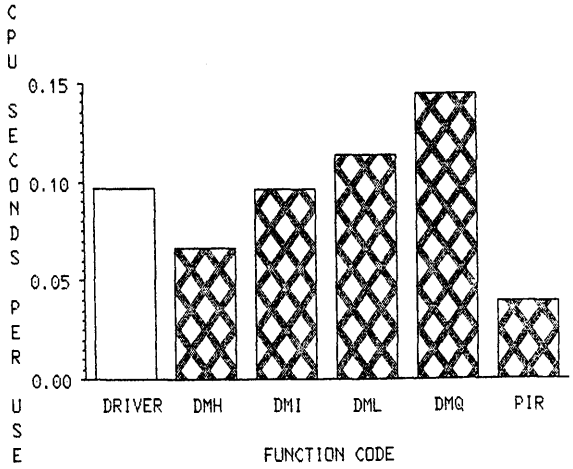


Figure 7. IBM 3090-400 CPU Seconds Per Function Code

Response Time Breakout

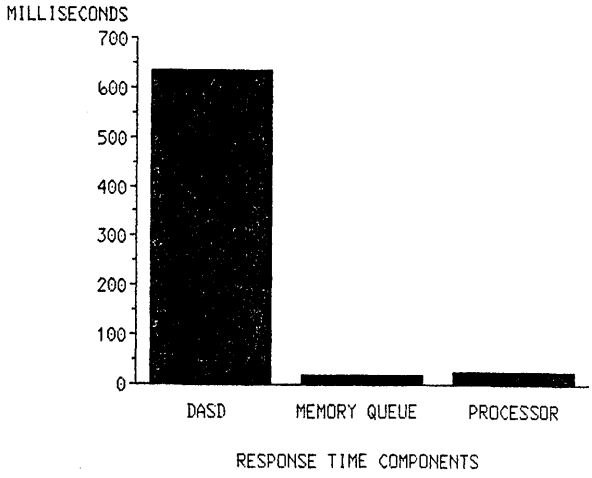


Figure 8. Driver Response Time Components

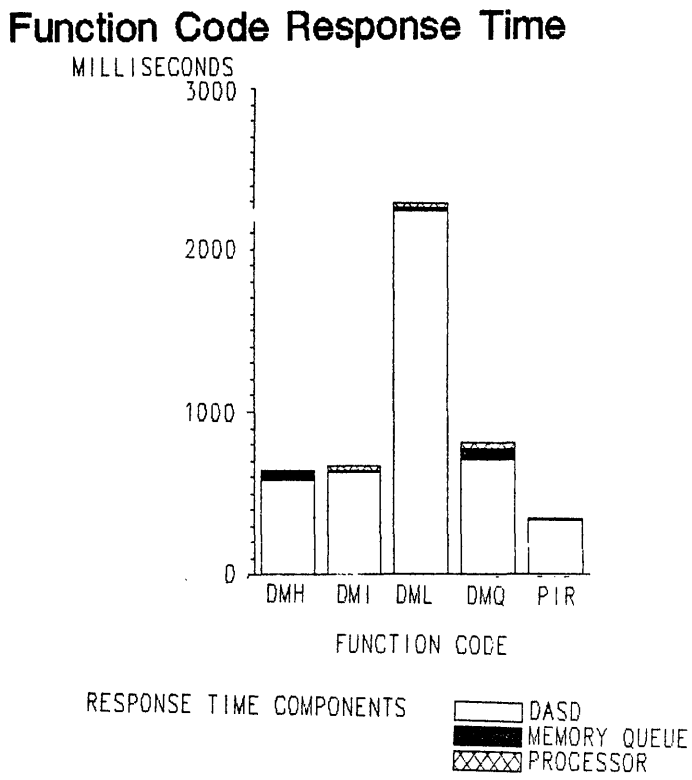


Figure 9. Function Code Response Time Components

Device Utilization At Module Level

WORKLOAD: IMS_DLMP100_DMI DEVICE: LDB101

FILE	USAGE
FSAPHLDD_VDAM	0.8%
FSAPLOGD_VDAM	0.4%
FSAPPDRD_VDAM	12.2%
TOTAL=	13.3%

Figure 11. Device Utilization by File by Function Code

Response Time Breakout
Device Level

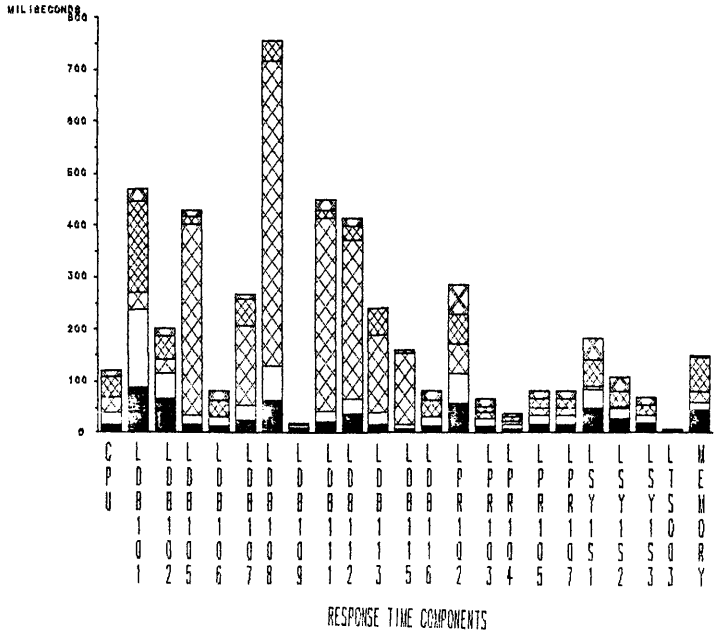


Figure 10. Function Response Time At Device Level

IMS CPU Usage - 3090-400

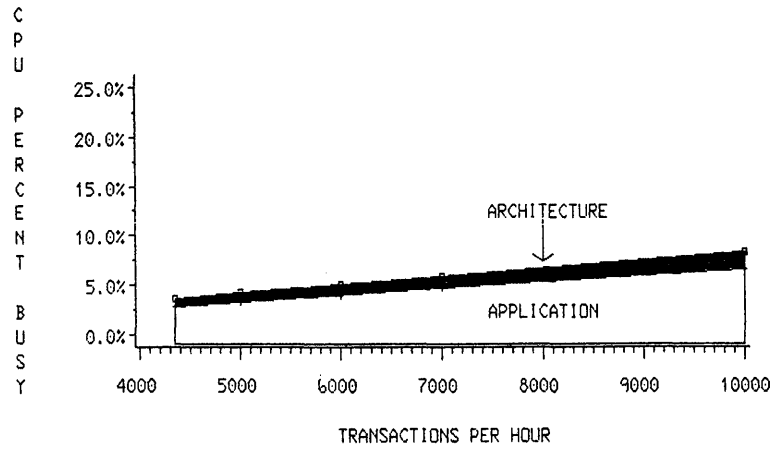


Figure 12. CPU Requirements Based on Transaction Growth

IMS Response Time Performance

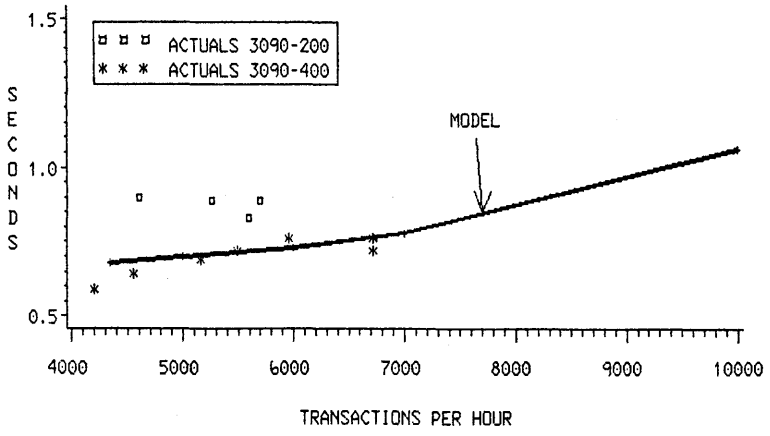


Figure 13. Model Valiation

MEMORY MANAGEMENT FOR LARGE IBM SYSTEMS

Michael R. Lee
Candle Corporation

ABSTRACT

Recent hardware offerings from major processor vendors are providing an amazing increase in real storage capacities. For most users of these processors, this increase provides much needed relief for overloaded paging subsystems. For those who constantly push the limits of hardware technology, however, these larger memories are still causing problems with application performance. This paper will explore some of the recent trends in increased real storage capacities together with some of the other trends - specifically expanded storage, and discuss a coordinated analysis methodology which may be used to "tune" the available memories for optimum performance and capacity.

INTRODUCTION

Memory management in the Multiple Virtual Storage (MVS) environment has been a challenge for system programmers and capacity planners for many years. Typical approaches to memory management center around controlling paging and swapping through use of storage isolation and appropriate Installation Performance Specification (IPS) definitions of performance groups and domains. With the arrival of IBM 3090 processors and Expanded Storage, the challenge of effective memory management has become even more difficult.

This paper presents a memory management methodology that has proven to be useful in a number of MVS installations in the USA. The topics discussed include: Hierarchy of memory management, System Resources Manager (SRM) memory control philosophy, Dynamic memory controls Categories of memory, Real memory management, Expanded memory management, and Auxiliary memory management.

HIERARCHY OF MEMORY MANAGEMENT

In a virtual storage environment, a task must have currently executing instructions in a frame of real memory to enable the CPU instruction fetch hardware to access the instruction to be executed. If the instruction to execute is not in a real storage frame, a page-fault is recognized triggering the page-in process. The specific page must be moved from an auxiliary storage location (auxiliary storage or expanded storage) into an assigned frame of real storage. From a task performance viewpoint, page-fault processing is non-productive time for the task because it must wait for the page to be retrieved from auxiliary storage. Too many page-faults for a task mean too much time waiting for page I/O and not enough time executing application program instructions.

The number of pages for an application which must be in real storage to prevent page-in waits from becoming a performance bottleneck can be called the "critical working-set". Maintaining or protecting the critical working-set for active tasks is the overall objective of memory management.

Unfortunately, real storage is a finite resource and is shared among all active tasks. Paging is a result of multiple tasks competing for a limited pool of real storage frames. Some tasks are more important than other tasks. The important tasks are favored or preferred both in access to/ownership of real storage frames and in dispatching priority sequence.

Memory management requires careful planning and periodic monitoring. Planning includes selection of appropriate IPS parameters to allow enough separation of work loads among performance groups, selection of appropriate domain constraints (MPL boundaries), and dispatching priorities. Planning also includes design of the paging subsystem (even if expanded storage is present).

Periodic monitoring is necessary to observe possible shifts in work load mix. Work load mix variations may result in an increase or decrease in average system wide MPL. Changes in MPL indicate there are more or fewer address spaces competing for available real storage frames. Failure to identify these shifts in work load patterns may result in overloads of domains or possibly artificial constraint. The domain definition process should include an analysis of the storage/paging requirements for the work load based upon expected MPL levels. When this level is determined, appropriate constraint values for the domain are assigned to avoid unnecessary memory overload or underload.

Memory management must be structured in an hierarchical fashion. This philosophy includes consideration that the load on a modern MVS system will fluctuate dramatically during the processing day.

The purpose of memory management is to keep critical application pages in real storage and if necessary, remove pages for non-critical applications. Thus as contention for real frames increases, stealing will occur from low priority tasks.

The actual stealing of frames is performed by RSM (Real Storage Manager), but SRM (System Resources Manager) determines which address space(s) will have frames stolen. SRM also determines how many frames are stolen from each candidate address space. The stealing process may be altered by specification of storage isolation for a performance group. The performance groups are defined in IEAIPSnn.

If real storage frame contention becomes critical, SRM will consider swapping out some tasks in low priority domains. This event triggered swapping occurs under control of SRM and is influenced by user selected options set in IEAOPTnn. The determination of low priority

is made by SRM through an analysis of the contention indices for all domains.

Paging is not bad in and of itself - paging is what makes virtual storage systems work. Most programs have very definite instruction reference patterns. The subset of pages within an address space that correspond with the reference pattern are the only pages that need be resident in real storage.

The trick is to keep the "necessary" pages in storage for active address spaces and to remove idle or inactive pages to make room for more active pages. Unfortunately, the inactive pages that have been removed from real storage frames may and often are re-referenced by the owning address space. Page-delay time is the time required to determine the location of a page (in expanded storage or on auxiliary storage) and to relocate it back into a real storage frame. Page-delay time now becomes a significant component in the performance of the address space, especially if the address space is processing a trivial interactive request.

Tuning choices involve selection of parameters to tell SRM how to manage memory. You must determine how much paging is acceptable for a particular address space (e.g. CICS, IMS, TSO, etc.) and set up a memory management philosophy that will keep paging and thus performance for key applications within acceptable service level bounds.

The tuning choices available are: storage isolation, MPL constraints for domains, overall system MPL, UIC level, page-fault rate, demand paging rate, and others. These options are discussed later in this paper.

SRM MEMORY CONTROL PHILOSOPHY

Many people expect SRM to bail them out of a performance problem resulting from ineffective option choices (or default parameter values).

SRM is just a group of programs that attempt to deal with resource overload situations such as CPU 101% busy, high paging rate, ASM slot shortage, low UIC, etc. SRM is a complex component that includes many algorithms.

The algorithms are influenced by option selections in IEAOPT and IEAIPS. Depending upon which options are selected and the values used, SRM control can be described as absolute or dynamic.

Absolute controls are analogous to a sledge hammer. The system runs unrestricted up to the point of overload, then SRM must take drastic measures to reduce the load demand (i.e. swap out everybody I can). The system then thrashes by trying to exchange swap the active users with the swapped users.

Dynamic controls are analogous to screw drivers. Dynamic controls are fine tuning mechanisms that allow gradual adjustment of the MPL to reduce thrashing and improve throughput and response time.

Absolute memory controls are the outside bounds controlling the number of tasks allowed to compete for system resources. The MVS MAXUSER parameter defines the absolute maximum number of address spaces that MVS may create (started tasks + TSO + batch). MAXUSER parameter sets the maximum number of TSO address spaces that MVS may create.

Once an address space is created, access to real storage frames (and thus the CPU) is controlled through the MPL. There are two levels of MPL to consider. First is "system wide MPL" which represents all address spaces now in real storage and competing for resources. Secong is "domain MPL" which represents the current MPL for a specific domain. Each domain has a domain MPL value. Each domain's MPL value is adjusted by SRM based upon the system wide MPL and the contention index for each domain.

MPL adjustment depends upon the IPS options selected for the performance group (i.e. the domain to which an address space is assigned - which can change for multiple period performance groups).

The domain is the primary point of control over a particular work load by limiting how many address spaces of this type are active at a time. Constraints for a domain are evaluated with respect to the memory, CPU, and I/O requirements of the work load(s) running within this domain.

IEAOPTnn parameters are used to provide SRM with performance boundaries for critical system contention indicators and thus affect system-wide MPL based upon SRM's observations of these indicators. The IEAOPT parameters that affect MPL adjustment are listed below:

RCCPTRT: Page-fault rate threshold
RCCUICT: UIC - measure of how active frames are
RCCASMT: ASM queue length
PAGERT1: Demand page rate threshold for 1 CPU on-line (also applies to more than 2 CPUs)
PAGERT2: Demand page rate threshold 2 CPUs on-line

Work flow through an MVS system varies greatly during the processing day. There are times when real memory and CPU become overloaded. The memory management question now becomes: "how should SRM deal with this situation?"

RCCPTRT and RCCUICT provide a mechanism for establishing the bounds of "acceptable" real storage contention as measured by UIC and page-fault rate. If UIC or page-fault rate exceed the upper bounds, system wide MPL will be reduced. Of course, this means SRM must now select the domain with the lowest contention index that is not at its minimum constraint and swap an address space out. Clearly, this action will reduce the number of competing address spaces and thus should reduce storage contention.

RCCASMT, PAGERT1, and PAGERT2 are very volatile numbers. Using these values for MPL adjustment can lead to swap-thrashing if the values used are not carefully selected and monitored. In most cases, these parameters are set to very high values to eliminate their influence on MPL adjustment.

Adjusting MPL based upon UIC and page-fault rate is usually a sound approach to MPL management. This is because these measures tend to be fairly consistent values over time thus avoiding radical changes in MPL due to temporary spurts of paging activity.

NOTE for 3090 expanded storage users: Be aware that demand paging done to expanded storage is not included in paging statistics and is not included when SRM checks these OPT thresholds. Only pages sent directly to auxiliary storage and page reclaims are counted.

DYNAMIC MEMORY CONTROLS

The objective of dynamic memory management is to select a combination of OPT/IPS parameters which will allow SRM to protect response critical tasks to the point where service level objectives are being met.

Selecting options which allow SRM to adjust the degree of protection to that required to meet service level objectives helps avoid needlessly tying up memory resources for a task that does not need them. A good example of the dynamic management approach is to specify the private-page-rate parameter (PPGRT/PPGRTR) when using storage isolation (PWSS).

Of course, one must realize that there are always exceptions to the rule and as such, it may be necessary to "waste" some amount of storage in order to provide an address space with sociopathic tendencies enough real storage resource to meet service level objectives (usually CICS, DB2, IMS etc.).

CATEGORIES OF MEMORY

There are three categories of memory addressed in this paper. First is real storage. Second is expanded storage and third is auxiliary storage.

As processor speeds have increased, the need for more memory has increased. This relationship exists because instructions must be in real storage before the processor can execute them. A faster processor can execute a series of instructions at a given speed, but most programs perform some type of I/O operation during their processing. I/O processing is extremely slow when compared to CPU instruction cycle times (e.g. if one nano-second equaled one second, one milli-second would equal 11.5 days). This time differential requires a fairly large queue of work awaiting execution to keep the CPU(s) busy - this means a fairly large MPL is required - and large MPLs require large amounts of real storage.

Expanded storage provides a mechanism to increase the availability of real storage frames (thus allowing even higher MPLs) by requiring fewer pages to be kept in real frames. If a stolen page is re-referenced, it can be quickly retrieved from expanded storage (on non-expanded storage processors, these page references resulted in comparatively slow page data set I/Os).

Auxiliary storage has been around for many years now and has not changed much in physical implementation (i.e. page data sets required, swap data sets optional). There have been some interesting changes in management algorithms however. Changes such as the contiguous slot algorithm (block paging capability), suspend-resume channel program, etc. greatly improve the efficiency of paging, but the most significant shortcoming still exists and that is the time required to do physical I/O to a rotating disk device.

REAL MEMORY MANAGEMENT

The objective of real memory management is to keep active pages in real storage frames as a result of careful MPL management and a reasonable demand paging rate.

Absolute real memory controls are rigid and inflexible. Three examples of absolute controls are: V=R, Page-fixing, and Non-swappable. Once set, these controls provide no mechanism for SRM to adjust memory allocations based upon the demand for real storage frames.

Dynamic real memory controls are flexible (within user defined bounds) allowing SRM to dynamically adjust memory allocations based upon demand for real storage frames. Logical swap is an example of a dynamic real memory control.

Dynamic controls are effective if used properly, but are expensive if abused. Inappropriate or excessive use of dynamic controls can result in both ineffective resource utilization and limited resource availability for other address spaces (i.e. you may improve the targeted application but degrade every other address space in the process).

Page stealing/faulting is one of the most serious performance problems faced by large interactive address spaces. Page stealing is serious because should the address space re-reference the stolen page, a page-fault will occur which will stop the task's TCB. If the TCB suffering the page-fault is the primary or major work TCB (as in CICS or TSO or IMS control region), all processing stops for that application and cannot resume until the referenced page is returned to a real storage frame.

The simplest method to prevent paging is to run V=R mode. This is no longer practical considering the large MPLs run on modern processors. The next method is to affect the page stealing process through SRM parameter options such as storage isolation.

NOTE: For 3090 expanded storage users: Storage isolation no longer guarantees a specific resident working set (real frame working set). Storage isolation now computes the number of real frames occupied and adds the number of expanded frames occupied to arrive at "working set size" to which the storage isolation target working set size is compared. Expanded storage frames that exceed the target working set are then eligible for migration to auxiliary storage should expanded storage become constrained.

LOGICAL SWAP

Using IEAOPT parameters, the system programmer can provide SRM with a lower and upper bound for each of the IEAOPT parameters affecting logical swap. Logical swap is expensive in that idle address spaces may continue occupying real storage frames with the expectation that the logically swapped address space will enjoy improved response time.

Logical swap demands real storage and if logical swap is permitted during periods of severe real frame contention, a few address spaces will see improved response times but will likely result in higher demand paging rates for all other address spaces. The effect on other address spaces may be acceptable in a large TSO environment where TSO is the primary application.

Logical swap is most effective if implemented as a dynamic facility. SRM may adjust the duration of the logical swap based upon real storage frame contention. If contention is high, the duration of logical swap will be short (in fact, SRM may be allowed to "shut off" logical swap through selection of appropriate IEAOPT parameters). If contention is low, the duration of logical swap will be longer. The IEAOPTxx parameters used to control logical swap are listed below:

LSCTMTE LOW and HIGH Think Time Thresholds (SECONDS)
LSCTUCT LOW and HIGH UIC Thresholds (SECONDS)
LSCTFTT LOW and HIGH FIXED below 16MB real Storage (%)
LSCTFET LOW and HIGH FIXED/ALLOCATED Real Storage (%)
LSCTAFQ LOW and HIGH AFQ Thresholds (FRAMES)

The most common wait reason for a TSO user is "terminal-input" wait. This wait indicates the previous command has completed and the address space is waiting for the user to enter the next command on the terminal (thus terminal-waiting-for-input). The reasoning behind this is that an interactive user will spend some amount of time "thinking" between transactions. The user may "think" for one second or two hours, SRM has no way of telling how long the wait might last.

Logical swap allows a user to remain in real storage for a time interval and if the wait time exceeds this value, the address space will be converted to a physical swap thus freeing the real storage resource. The objective is to avoid swapping an address space that may shortly become active again.

NOTE: Swap-trim occurs for an address space entering logical swap status. This trimming can reduce the effectiveness of logical swap should the trimming lead to an excessive number of subsequent page-faults.

NOTE: For MVS/XA 2.1.7 users, an address space targeted for swap to expanded storage may, at the discretion of SRM, be logically swapped EVEN THOUGH LSCTMTE IS SET TO (0,0). SRM evaluates the contention for real storage and if under-utilized, will force logical swap. SRM will also reduce the degree of swap-trim for a logical swap candidate. If the logical swap becomes a physical swap, the balance of swap-trim will be performed before creation of the secondary working set.

NOTE: For MVS/XA 2.1.7 users with expanded storage - If using application packages other than ISPF under TSO, ECB wait may become the most common wait reason (rather than terminal-input). This is significant in an expanded storage environment if using the IEAOPTnn parameters that affect the destination of working-sets. ECB wait TSO address spaces are treated as part of the non-privilidged, non-terminal wait group which includes batch. Serious delays in swap-in may occur in high storage utilization situations.

Storage Isolation provides a mechanism for the system tuner to influence paging within an address space. This influence is based upon preventing page-stealing (including swap-trim) from occurring below a limit specified in PWSS as defined in the performance group definition for this address space.

Clearly, if the address space is large, preventing frame stealing from this address space could lead to more severe stealing occurring in other "non-protected" address spaces. Storage isolation does not increase the real storage capacity of a processor, it only influences the distribution of those frames among active address spaces.

Logically, if SRM/RSM are stealing pages from an address space and those pages are re-referenced soon after stealing the frames in the first place. Storage isolation is an effective tool in minimizing this type of page thrashing.

Certain address spaces (e.g. CICS, IMS, DB2, etc.) are performance critical applications and due to transaction volumes, normally cannot tolerate private area page rates above 5-10 pages per second without causing serious delays in transaction processing. These delays from suspension of the TCB in control at the time of the page-fault until the page is returned to a real storage frame. In this type of address space, protection is extended beyond the case where recently stolen pages are referenced to a point where most all pages, regardless of age, are protected. This is often necessary due to the very large working set sizes of this type of address space.

Storage isolation is a very powerful option and therefore should be used sparingly. It should only be used in specific situations where paging is causing performance degradation for an application.

Before using storage isolation, be sure there are no performance problems in the paging subsystem itself. Look for path contention, number and type of page data sets, over initiation of the system (too many active tasks), etc.

Remember, it is not necessary to stop paging altogether! The objective is to reduce paging to the point where the application will begin meeting its response time objective. The next question is: "how much storage isolation is enough?" It depends. The limits for PWSS should be high enough to do the job, but low enough to avoid impacting other address spaces. Thus the appropriate values must be determined through trial and error.

As a place to start, use a real time monitor to observe the working set size and page-fault rate for the problem region when service level objectives are being met. Repeat this observation for the region when service levels are not being met. Compare the two observations and select PWSS and PPGRT/PPGRTR parameters that appear appropriate. Implement and monitor - repeat until desired effect is obtained.

Storage isolation may also be used for swappable address spaces to alter the page stealing algorithm at the point of swap-trim. This allows the swap-out set size and thus the swap-in size to be controlled.

Storage isolation allows one to define a "working-set size" for the common area. Often, system tuners overlook the common area when implementing storage isolation. This can be a serious oversight. If the working set for CICS is protected in an effort to improve response time, but doing this places stress on the common area (common area frames are eligible for paging), we could end up with serious paging problems in the VTAM communications buffers and/or in PLPA. Page-faults in the common area can be even more impactive on response time than the original private area paging problem.

Storage isolation can be used to protect pages, it can also be used to target specific frames for stealing. If a low PWSS threshold is specified for a large batch job, for instance, those frames in the working-set that exceed the PWSS threshold are preferred for stealing. This can be a means to direct stealing to particular address spaces to free frames needed by higher priority applications.

Storage isolation can be specified as a static value or as a dynamic value. Static values are specified by PWSS and CWSS but do not take into account the paging rate. Thus the protection is fixed at the lower limit of the specification. Dynamic values allow the woking set to fluctuate in response to the private page rate within the bounds set in PWSS/CWSS.

PPGRT, PPGRTR, and CPGRT are the parameters which allow a range of paging rates to be specified. Each option allows a lower limit and an upper limit to be specified. If the private page rate is below the minimum, the working set size will be reduced - but not below the lower limit specified in PWSS/CWSS. If the private paging rate rises above the upper limit, the working set size will be increased - but not above the upper limit specified in PWSS/CWSS.

PPGRT and PPGRTR should only be specified for non-swappable address spaces. Swappable memories usually are not resident long enough for a meaningful page/sec. rate to be computed (e.g. TSO). An address space must be resident and active for 30-60 seconds before a meaningful page/sec. rate can be computed.

PPGRT and PPGRTR differ in how the page/sec. rate is computed by SRM. For PPGRT, the page/sec. rate is based upon CPU seconds or execution seconds. For PPGRTR, the page/sec. rate is based upon residency time not execution time.

For cross memory address space, always use PPGRTR rather than PPGRT. This is recommended because most cross-memory address spaces spend little time executing under their own TCB structure. Cross memory callers to this type of address space execute the instructions under the caller's TCB. There is so little execution time accumulated under the cross memory address space's TCB that it is not possible to develop a realistic page/sec. rate for use by SRM in adjusting the working-set size.

It may be becessary to use PPGRTR for a volatile CICS system rather than PPGRT if there are large variations in transaction arrival rates to CICS. The CICS region may sit "idle" for long periods and thus accumulate little execution time. In a seriously constrained real storage environment, this could result in the CICS working-set being reduced thus causing a rash of page-faults when the next burst of transaction activity arrives.

With PPGRT, the working-set will be adjusted more quickly (i.e. will adjust more rapidly to changes in execution time) than PPGRTR. If the private page-rate goes up quickly, PPGRT will cause SRM to increase protection quickly and if the rate drops quickly, SRM will decrease protection quickly.

If using PPGRTR, since the page-rate is computed over residency time rather than execution time, spurts of paging activity or rapid drops in paging activity will be averaged out over residency time and thus cause the working-set adjustment both increase and decrease to occur more slowly.

If PWSS is specified alone (PPGRT or PPGRTR not specified), SRM sets the target working-set size set to the minimum value. Page stealing may occur within the address space down to this level. Therefore, the "b" value is ignored.

If PWSS is specified together with PPGRT or PPGRTR, the working set size "floats" between low and high based upon the private page rate within the address space.

If swap-trim is reduced, the size of the working set at swap-out will increase. Reduced swap-trim also means the swap-in working set will be larger and thus take longer to physically swap back into real storage. The trade off is between spending more time swapping the working set back into real storage and spending time waiting for individual page-faults to be processed after swap-in. In most cases, it is better to suffer the slightly higher swap-in time rather than individual page-faults. The objective should be to keep trivial transaction page-fault rates below 5 per transaction (2 seems to be a good ROT number).

NOTE: For 3090 expanded storage users: Storage isolation no longer guarantees a specific resident working set (real frame working set). SRM now computes the number of real frames occupied and adds the number of expanded frames occupied to arrive at "working set size" to which the storage isolation target working set size is compared. Expanded storage frames that exceed the target working set are eligible for migration to auxiliary storage should expanded storage become constrained. Since we can no longer control resident set size, it seems we are to believe that page-faults from expanded storage are of no concern since they only take 70-75 micro-seconds compared to 15-20 milli-seconds for auxiliary storage.

Before using storage isolation, be sure there are no performance problems in the paging subsystem itself. Look for path contention, number and type of page data sets, over initiation of the system (too many active tasks), etc.

EXPANDED STORAGE MANAGEMENT

Expanded storage is managed by RSM under control of SRM. Pages move from real to expanded storage under control of RSM but only after checking with SRM to see if it is OK. Pages move from expanded storage to real storage under control of RSM. The page-fault interrupt handler receives control after the PIC 11 and gives control to RSM who determines the location of the page (could be in expanded storage or on auxiliary storage). If in expanded, a synchronous page move is done to physically move the page from expanded to real. Clearly, this process consumes CPU cycles. The time required to resolve a page-fault from expanded storage is estimated by IBM to be 75 micro-seconds for standard 3090 models and 70 micro-seconds for 3090-E models. Of this time, approximately 30 micro-seconds is spent actually moving data. The other 45 micro-seconds are spent in the interrupt handler and RSM code.

Swapping is also performed (under SRM control) to expanded storage and again, RSM must perform the page movement operations required to move the working set into expanded storage. Note that there is no "block-paging" capability for expanded storage. Each page in a swap operation is moved individually. There is a cross over point where the CPU overhead to swap an address space from auxiliary storage is less than that for a similar swap from expanded storage.

As with any hardware/software feature release, we must look at what expanded storage offers, but also how it affects the existing knowledge base for tuning procedures and capacity planning.

A group of new IEAOPT parameters has been added to provide some input about how we would like SRM to manage expanded storage frames. These OPT parameters work in conjunction with the Criteria Age Table and Migration Age.

The criteria age table allows SRM to regulate direct use of expanded storage through determining what type of pages are sent into expanded storage. The criteria age table values are changed through use of the IEAOPT expanded storage parameters. These parameters allow decisions to be made about where a particular address space should be swapped: to expanded storage or directly to auxiliary storage?

Unfortunately, these parameters only affect swapping decisions if expanded storage becomes constrained (i.e. migration age drops below 32,767). Therefore, you cannot absolutely determine the swap destination for a particular type of address space.

Migration is the process used by RSM and SRM to manage expanded storage frames. The primary purpose is to maintain a supply of available expanded frames for use by RSM for page-outs and swap-outs. Migration describes the process of determining how long a page remains in expanded storage and thus determins when old "inactive" pages are to be sent to auxiliary page data sets. Migration is not controlled by IEAOPT parameters. It is dynamically controlled by SRM and is adjusted based upon demand for expanded frames.

NOTE: The channel subsystem cannot access expanded storage, so migrating pages must be moved back into REAL memory before ASM can page them out to auxiliary.

Some old tuning terms have changed meaning with the addition of expanded storage. Working set now represents the total of frames occupied in real storage and the frames occupied in expanded storage. This dramatically changes how storage isolation functions. Storage isolation no longer holds frames in real storage.

A Page fault is not counted if resolved from expanded storage. It is only counted if a reclaim or physical I/O to auxiliary storage was required. This affects all the OPT parameters that trigger on demand page rate or page-fault rate.

Migration age is supposed to give you some idea about how long a typical page will remain in expanded storage before migrating to auxiliary. The stated purpose of migration age is to function in a manner similar to UIC for real storage management, but unlike the page frame table which contains a counter for UIC, there is no equivalent counter for an individual expanded storage frame.

There are three types of migration that can occur in expanded storage systems. First is LRU migration. LRU migration is triggered by a shortage of available expanded storage frames. Second is Re-configuration migration which is used to move pages out of expanded storage frames which are being varied off-line. The third type of migration is Purge migration. This is the process of moving the secondary and/or primary working set of an address space from expanded frames to auxiliary storage. Neither re-configuration nor purge migration alters the migration age. Therefore, you may see migration occurring from expanded to auxiliary storage while a very high migration age exists.

For an address space that is actively executing in real storage, the working set will consist of a residence set which is the current number of real frames occupied. All other pages that belong to this address space will be either in expanded storage or on auxiliary storage and are considered to be "non-working set pages".

For an address space that is to be swapped to expanded storage, the residence set (those frames surviving swap-trim) will be divided into two parts: primary working set and secondary working set. This division is performed by RSM as part of the swap process, but is only performed for swaps to expanded storage. Direct swaps to auxiliary storage are performed without division. The primary and secondary working sets are then moved into expanded storage if SRM approves (i.e. if there are enough expanded frames available to contain the swap pages).

The address space now has three distinct parts: primary working set, secondary working set, and non-working set pages.

The primary working set consists of: LSQA, Fixed frames, and a single frame for each segment used by the address space. The secondary working set consists of: All working set pages not in the primary working set including storage isolated frames. Non-working set pages are what ever is left over.

Swap migration is a multi-step process. If, during LRU migration processing, RSM finds that the next page to be migrated from expanded storage frame is part of the primary or secondary working set for an address space (a bit in expanded storage table entry indicates this fact). RSM then asks SRM for a special type of swap (migration swap) which will be used to send the remaining portions of this address space to auxiliary. First, the secondary working set pages will be sent to a local page data set. Then the primary working set will be sent to a swap data set (if there is one) or to local page data set.

When the address space becomes active again, it must now be swapped in from auxiliary storage using a two-stage swap-in process. The primary working set comes in first. The address space is now dispatchable. Secondary working set pages revert to "non-working set" status and are retrieved through page-fault processing (shades of pre SP 1.3).

The implementation of expanded storage required some additional selection criteria be developed to help determine what type of address space should be sent to expanded storage and what type to send directly to auxiliary. There are three classes of users: Non-swappable/Common area/Privileged, TSO Sessions in Terminal Wait, and all others (other TSO, batch, etc.).

These classifications allow distinctions to be made about how to handle types of page operations based upon resource contention. Thus when real and expanded storage become constrained, "non-critical" address spaces may be sent directly to auxiliary and bypass expanded altogether.

For each class of address space, there are also classes of pages. This provides a mechanism for tuning expanded storage based upon type of page request and type of address space.

- SWAP Trim, changed pages
- SWAP Trim, unchanged pages
- SWAP Working Set (TSO terminal input)
  primary/secondary working set
- PAGE-OUT, unchanged pages demand page-outs
- PAGE-OUT, changed pages
- STEAL, changed pages page steals
- STEAL, unchanged pages
- Virtual fetch paged virtual fetch

AUXILIARY STORAGE MANAGEMENT

Auxiliary storage is still with us - even with expanded storage. The old rules of thumb still apply when it comes to paging subsystem design. The number of page data sets required is based upon the volume of page

movement to auxiliary. The number of volumes must then be spread over the I/O subsystem in such a way as to provide parallelism through channel paths and devices.

Additional configuration choices include the type and combination of page data sets - whether to swap to local page data sets or include swap data sets and use of directed VIO to separate VIO paging from demand/swap paging.

The introduction of the seldom ending channel program in MVS/SP 1.3 was an interesting approach to reducing the Start I/O overhead for ASM channel programs. Suspend/Resume provided the ability to "suspend" a channel program and resume it at a later time using a fast-path line through IOS. This worked well, except that the suspend status was stored in the sub-channel for the page device. This means that as far as IOS is concerned, the device is still busy.

If another non-ASM I/O request is made to the volume, IOS informs ASM that someone else wants to do I/O to the device and ASM must terminate the suspended channel program. IOS may them allow the non-paging I/O to begin on the device. This situation should be avoided - especially in moderate to heavy paging situations where this handshaking can be very expensive to paging operations.

In addition, if you are running MVS/XA, you must consider interrupt subclasses. There are now two categories of I/O interrupts. The first is paging, the second is non-paging. Paging interrupt subclass is associated with the UCB containing a page data set. However, since the association is through the UCB and not the data set, all I/O interrupts occurring on this device will receive the paging interrupt subclass (even if for a non-page data set).

Avoid allocating non-page data sets on page volumes - and if you do, don't access them during peak paging periods. Specifically, avoid placing INFO/MVS, SMP DLIBs, etc. on the back end of a page volume.

SUMMARY

- Paging is a design feature of virtual storage
- Don't expect SRM to fix severely over-committed memory
- Memory management should follow dispatching priorities
- Avoid using absolute controls
- Utilize dynamic memory controls
- Logical swap - last use for "extra" memory
  (extend use for expanded storage)
- Use storage isolation carefully - diminishing return
- Don't use Storage Isolation to compensate for other problems in paging subsystem

BIBLIOGRAPHY

GC28-1149 MVS/XA SPL Initialization and Tuning
SC28-0922 MVS/370 RMF Reference and User's Guide
GG22-9264 MVS Paging Performance Considerations
GG22-9346 MVS/XA I/O Performance Considerations
GG66-0213 IBM 3090 Model 200 Processor Complex Installation Notebook

FURTHER READING

LC28-1138 MVS/XA RMF Reference and User's Guide
LY17-5500 MVS/XA RMF Field Description and Analysis
LY28-1214 MVS/XA System Logic Library Volume 3
LY28-1242 MVS/XA System Logic Library Volume 10 part 1
LY28-1243 MVS/XA System Logic Library Volume 10 part 2
LY28-1250 MVS/XA System Logic Library Volume 12

Modelling Memory constrained systems

Gilbert Houtekamer and Rob van Hoboken
Delft University of Technology
Faculty of Electrical Engineering
4 Mekelweg
2600 GA Delft
The Netherlands

Guus Bonnes*
IBM IBS-SC
Boerhaavelaan 11
2713 HA Zoetermeer
The Netherlands

ABSTRACT

The System Resources Manager (SRM) component of the IBM MVS operating system controls the allocation of resources to the various users of the system. In this paper the memory control algorithms of the SRM are studied, and a model is proposed. The model distinguishes between different operating conditions of the system, reflecting the behavior of SRM control as the memory demand increases. Results are presented for a single class time sharing model. Some suggestions are given for measurement procedures, and for a multiple class model. The difficulties in collecting the model and workload parameters precluded model validation. In its current form, the model can be used to gain a better understanding of the SRM memory control algorithms.

1. Introduction

Modeling memory constrained systems is difficult, since not only the CPU and (non-page/swap) I/O load must be quantified, but also the paging and swapping load. The paging and swapping load depends not only on the characteristics of a transaction or job, but also on the system load: as contention for memory increases, the paging load will increase as well. On the top of that, the operating system will adapt its behavior to the memory contention it observes.

In this paper we will describe a model that includes memory contention as indicated by both paging and swapping. Although the model was primarily designed to match the memory control algorithm of SRM in an IBM MVS system, a similar methodology can be applied to other systems.

Section 2 of this paper describes the operation of the SRM's memory control algorithms and introduces the terminology used in this paper. Section 3 describes a model for these algorithms. In section 4 results are presented for a single class model. Section 5 discusses extensions to a multi-class model. Some measurement and validation procedures are evaluated in section 6. Finally, section 7 summarizes the results.

2. SRM Memory Control

In a large time sharing system many terminal users and batch jobs have to compete for the system's resources. In an IBM MVS system, it is the responsibility of the System Resources Manager (SRM) to allocate the resources in the system to its users, in such a way that the throughput is maximized and/or that each user receives the service he is entitled to. The resources controlled by SRM include the processor, the I/O devices and channels, and main memory. In this paper we will focus our attention to the main memory control algorithms. In most of the discussions in this paper, it is assumed that no I/O or CPU bottlenecks exist.

The MVS unit of memory allocation is a page. For each active task, only a certain number of pages are allowed to be in main memory. The pages kept in main memory are those that were referenced most recently, since they are most likely to be referenced again. In this paper the term *working set* will be used to refer to the pages in main memory for a task; the size of this operational working set thus depends on the system load [cf Denning 1980]. SRM adjusts the working set size for each task in main memory, based on the *Unreferenced Interval Count* for the individual pages.

A UIC is maintained for each page in main memory (frame in MVS terminology) indicating how long it has been since a task referenced it [IBM 1985]. Each second, SRM will increment the UIC counter for each frame that has not been referenced during the last second[1]. If it is necessary to make room for more tasks, SRM removes from all active tasks those pages that have their UIC equal to the highest UIC in the system, until sufficient room is available. This guarantees that the age of the oldest unreferenced page is approximately the same for all tasks in the system [IBM 1982].

In an unconstrained system, the UIC for each task will be high, since there is no need to remove old pages before termination of the task. As the load goes up, it will more frequently be necessary to remove the oldest pages (with the highest UIC). This results in smaller working sets, making room for more tasks, but also increases the probability that a page referenced is not in main memory. Such an event is called a page-fault; the task will continue its execution as soon as the page is loaded into memory again.

In a time sharing environment the users spend most of their time thinking and typing. They only consume system resources when their transaction is active. If sufficient memory is available to hold the pages for both the thinking and the active users, there is no need to load these pages into main memory for each transaction (swap-in), and to remove them each time a transaction is completed (swap-out). Instead, SRM performs a *logical swap*: the user is deactivated but kept in memory when a transaction completes. If sufficient memory is unavailable, a *physical swap* is performed for (some of) the users: the user pages are written to external memory when a transaction completes.

1. In MVS/XA this is not strictly true, as the update of the UIC is less frequently done if the memory utilization is low.

* Although the author is currently employed by IBM, the opinions expressed together with the recommendations made are strictly his

SRM bases this *swap decision*, i.e. whether to perform a logical or physical swap, on the *system thinktime* and the observed user thinktime from the transaction just completed. If the user thinktime exceeds the system thinktime, a physical swap is done. If not, a logical swap occurs. The idea is that users with a short thinktime will have their next transaction ready soon, so it is beneficial to keep them in memory. Essentially, the memory allocated to logical swapping is used as a cache for the most active tasks. The system thinktime is derived from the UIC: every 20 seconds the SRM will increase the system thinktime by 0.5 second if the UIC exceeds 30, decrease it by 1 if the UIC is less then 20, and keep it unchanged if the UIC is in the range 20..30[2]. The result is that the system thinktime (and thus the swap probability) is adapted such that the UIC is kept in the range 20..30. It will be clear that no control is possible as the system thinktime reaches zero, i.e. if all users are physically swapped for each transaction.

If SRM is unable to make room for more tasks by lowering the system thinktime because all users are physically swapped for each transaction, the UIC will further drop, until either the CPU is saturated, the paging rate becomes too high, or the UIC reaches 2[3]. When either limit is reached, pages are no longer stolen from active tasks, but entire tasks are removed from the main memory instead. Removing tasks will reduce the memory constraints and allow the UIC to go up. This process is called *exchange swapping*, since active tasks are swapped in and out memory. SRM controls the swap mechanism in such a way that each task receives its share of the total system capacity, in accordance with the SRM control parameters.

More swap mechanisms do exist, but they are not relevant to this discussion.

Figure 1 shows the memory layout for an MVS system under increasing load conditions, and the swapping that takes place in each case. Five different situations are discerned, in order of increasing memory contention. The figure shows the use of main memory for each situation, differentiating between *active users* (executing a transaction) and *logically swapped users* (in the think time). For clarity of presentation a single user class is assumed, consisting of terminal users.

Situation A in Figure 1 describes an unconstrained system: all user programs fit into memory easily. The UIC will be 255 (the maximum value), almost no paging occurs and no physical swapping takes place. Characteristic for this situation is that the memory is large enough to hold all tasks at their maximum working set size. The working set size in this system is defined as WS_{max}, since no page stealing takes place.

Situation B in Figure 1 describes a lightly loaded system, where some paging occurs since the working set is reduced to make all users fit in memory. No physical swapping occurs, since the UIC is still greater than 30.

Situation C in Figure 1 describes the normal load range where logical swapping is gradually replaced by physical swapping, based on the user and system thinktime. The UIC will vary between 20 and 30 in this area, since the system thinktime is used to control the swap decision, as described above. The working set size remains approximately constant in this situation, since the UIC is kept constant. The working set size for this area is referred to as WS_{normal}.
As more users are added to the system, the physical swap probability will increase to 100%.

Situation D in Figure 1 describes a heavily loaded system, where all transactions are physically swapped; the swap probability is 100%. Under these load conditions SRM will further reduce the working set, until the UIC reaches 2, the CPU is saturated[4], or the paging rate

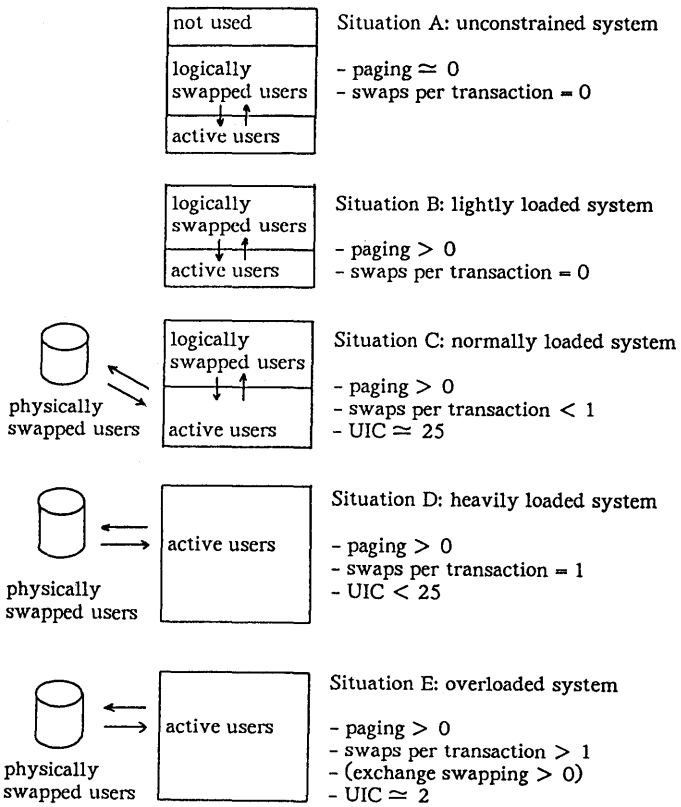


Figure 1. Memory layout for various system load conditions

becomes excessive[5]. The working set size corresponding to a UIC of 2 is defined as the WS_{min}, since SRM will no longer attempt to reduce it.

Situation E in Figure 1 describes the overloaded system condition: not all active tasks fit into the available memory, even if their working set is reduced to the bare minimum. SRM copes with the situation by starting exchange swapping: tasks are permitted to accumulate some service while in main memory, but are exchanged if their gain as compared to a swapped-out task exceeds some threshold. The active users are now partially kept in memory and partially waiting to be swapped in with an exchange swap. As more users are added to the system, the number of swapped out users will increase, without any increase in throughput.

2. 20 and 30 are the default thresholds; they can be changed using PARMLIB controls.
3. The definition of CPU saturation, high paging rates and low UIC can be changed using PARMLIB controls.
4. At least one task was waiting for CPU service during an entire SRM second.
5. The number of page faults per second is above a CPU dependent maximum value.

The values for some performance values and control parameters in the system under the situations described are sketched in Figure 2. The figure shows each parameter as a function of the system load. The UIC (shown in top) is the key control parameter for SRM; it is more or less constant in the unconstrained (A), normal (C) and overloaded (E) system conditions. The working set size is also constant in these situations, as shown with the second line. In the unconstrained situation (A) the UIC is constant at its maximum value. In the normal load situation (C) SRM controls the system throughput with the physical swap probability; the number of swap(-ins) per transaction rises from 0 to 1. In overload situation (E) exchange swapping is used in an attempt to reduce the memory constraints; the number of swaps per transaction now exceeds 1.

The system-wide paging rate will increase throughout A-D, since more users become active in memory (shown with the fourth line). Note that the system paging load in normal load situation (C) increases because more users are added to the system; the paging load for individual transactions remains constant, because the working set size remains the same.

In the overload situation (E) both the working set size and the number of swapped-in users remain constant as the load increases. The system throughput will decrease, since more time is required for the exchange swaps, as more users have to be served.

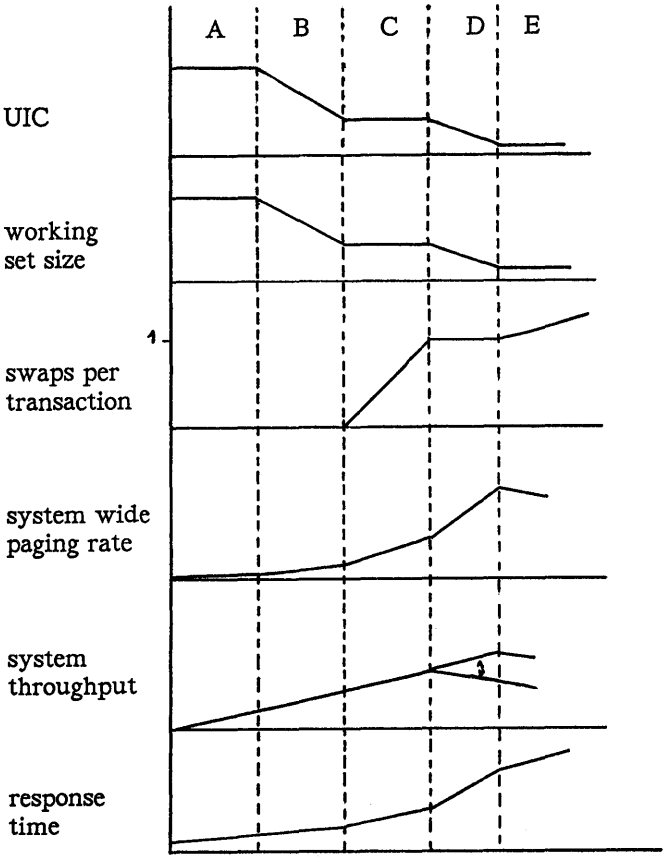


Figure 2. Performance and control parameters as a function of the memory contention (not to scale).

3. SRM Model

Each of the situations described above corresponds to a different behavior of the SRM control algorithms. The model described in this section reflects this: each situation is modeled separately.

Standard queueing models describe the workload of a transaction in terms of the CPU load and the I/O load to the various devices. For any model that should predict the paging and swapping load under varying load conditions these parameters are unsufficient: some information on the working set sizes and the corresponding page fault probabilities must be available.

In the model described it is assumed that for a given working set size, the probability that a page fault occurs can be computed (or is available from measurements), that is

$$P_{pagefault} = f\,(WS)$$

Furthermore, it is assumed that the working set size is a function of the UIC only, that is

$$WS = g\,(UIC)$$

The notion is that the UIC is a measure of memory contention in the system; for a given value of the contention index (the UIC), the working set size will be constant. This assumption is not strictly true, since the UIC counters are incremented each SRM second for all pages that have not been referenced during the last SRM second. That is, when only one task is active, it can execute about 1 million instructions during the SRM second. If n tasks are active, each task can execute only $1/n$ million instructions, and will reference less pages in its working set. The result is that in a heavily loaded system the UIC will be higher for a given working set size. In the model presented this effect is neglected. At first sight, this seems to be a problem with the model. You should realize, however, that any measurements performed in a real system will automatically account for this effect: a low UIC occurs when the load is high.

With these assumptions both the working set size and the number of page faults for a transaction can be computed from the UIC. One can also say that the relation between the working set and the page fault probability is known, and that the UIC tells at which point of the curve the system operates.

Looking at it this way, it is easy to see that other memory control policies can be described with a similar model, as long as the operating system maintains some kind of "memory contention indicator". For example, in a UNIX system one could use the page scan rate as a measure of contention [van Someren 1984]. In a VM system, the core table scan rate could be used [Beretvas 1986].

In the SRM model described below, the relations between the UIC, working set size and paging rate are used to compute the number of page faults per transaction, as a function of the number of terminal users. The model consists of a high level SRM part, which mimics the SRM control algorithms, and a low level submodel that provides response times and throughputs once the page and swap load are known.

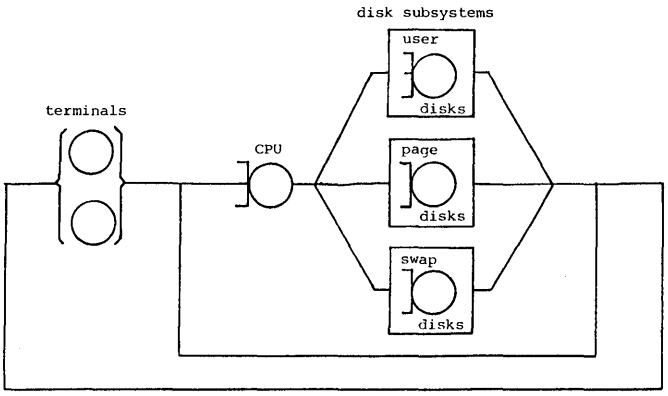


Figure 3. System submodel used

The system model used as a low level submodel is shown in Figure 3. It consists of a (variable) number of terminals, a CPU, and user, paging and swapping disks. The system model is solved using the Mean Value Analysis (MVA) method. For the disk drives, contention at the controller and channel level is modelled with a separate model, using the method proposed in [Lazowska, 1985]. This disk model computes the disk service time for a known throughput, initially estimated to be zero. The standard MVA method is then used with this service time estimate to compute a new throughput, to be used for a new disk service time estimate. The process is repeated until the throughput is sufficiently accurate.

For the purpose of this paper, the system model can be considered to be a black box, which accepts the number of terminals and the service demands for CPU (D_{CPU}), user I/O (D_{user}), paging (D_{page}), and swapping (D_{swap}) as parameters, and provides the system throughput and response time for these load conditions.

The algorithms used in the high level SRM model to compute the service demands for the low level model, are described in detail in the next subsections, for each of the situations A-E separately.

Basically, it is assumed that the SRM adjusts the UIC in situation B and D, the physical swap probability in situation C, and the exchange swap probability in situation E such that the tasks fill up the entire memory.

This is accomplished by setting the service demands for the system submodel, evaluating the submodel, and using the submodel results (particularly the overall response time and throughput) to obtain new estimates for the service demands.

The description below will be based on a single class model, with a time sharing workload (TSO).
The following notation is used:

$N_{terminal}$ the total number of users (active and thinking)

D_{CPU} the CPU service demand per transaction, including the overhead for paging and swapping

D_{user} the user I/O service demand per transaction for the non-page, non-swap disks

D_{page} the service demand per transaction for the paging disks

P_{page} the page fault "probability", expressed in the number of page faults per 1 million instructions

D_{swap} the swap demand per transaction for the swapping disks

P_{swap} the numer bo swap-ins per transaction

N_{active} the number of active users

$N_{outready}$ the number of users swapped out by exchange swapping

$WS_{\min}$ the working set size corresponding to the minimum UIC (2)

WS_{normal} the working set size corresponding to the UIC in the normal load situation (C)

$WS_{\max}$ the working set size for an unconstrained system (UIC large)

MemorySize the amount of main memory available to hold user tasks

3.1 Unconstrained System (situation A)

In situation A sufficient memory is available to hold all active and thinking processes. Since all tasks fit entirely in memory, the working set for each task is equal to $WS_{\max}$. The maximum number of users N_A accommodated in situation A is found as

$$0 \leqslant N_A \leqslant MemorySize / WS_{\max}$$

If the number of terminals $N_{terminal}$ is in this range, the working set size per task is known in advance: it is at its maximum value. The page fault probability follows directly from $P_{pagefault} = f(WS_{\max})$. The service demand for the paging disks per transaction can now be computed as

$$D_{page} = P_{pagefault} D_{CPU} = f(WS_{\max}) D_{CPU}$$

No swapping will occur in this unconstrained situation, so $D_{swap} = 0$. Since both D_{page} and D_{swap} are independent of the system response time and throughput; a single evaluation of the system submodel is sufficient.

3.2 Lightly Loaded System (situation B)

In situation B sufficient memory is available to hold all tasks, be it at a reduced working set size. SRM will steal pages from the active tasks, until all tasks fit into memory. The working set size reduction corresponds to a UIC reduction from the maximum UIC (255) to the "normal UIC", that is the UIC value in situation C (approximately 25). The system is in situation B if the number of users N_B is in the range

$$MemorySize / WS_{\max} < N_B < MemorySize / WS_{normal}$$

The working set size for a specific number of users $N_{terminal}$ is computed as

$$WS = MemorySize / N_{terminal}$$

From this working set size the page fault probability and the UIC (from the inverse of $WS = g(UIC)$, thus $UIC = g^{-1}(WS)$) can be computed. Note that the function g is a strictly increasing function for situation B, so the inverse function will exist. Like for the unconstrained system, no swapping occurs. Also, only a single evaluation of the system submodel is required since D_{page} and D_{swap} are again independent of the submodel results.

3.3 *Normally loaded system (situation C)*

In situation C sufficient memory is available to hold all *active* tasks. The working set size for each task is WS_{normal} since the UIC is approximately 25 over the entire interval. Since the UIC is constant, the number of page faults per transaction is also constant in this situation ($P_{page} = f(WS) = f(g(UIC))$). Since the number of transactions handled per second will increase as the number of users increases, the system paging rate will increase.

SRM controls the usage of memory through the physical swap probability; as more users are logged on, the probability for a physical swap will increase.

The system is in situation C if the number of terminals N_C exceeds $MemorySize/WS_{normal}$ and all active tasks fit in the memory, that is $N_{C,active}$ must not be larger than $MemorySize/WS_{normal}$.

In solving the model the physical swap probability P_{swap} is initially assumed to be one, i.e. all users are swapped ($P_{swap} = 1$). From the P_{page} and P_{swap} a first estimate for the service demands for paging and swapping can be computed. With these demands, the system submodel is used to estimate the number of active users N_{active} in the system. Multiplying N_{active} by the working set size WS_{normal} yields the memory used for active tasks. When the active tasks do not all fit into memory ($WS_{normal} N_{active} > MemorySize$), it is assumed that the system is in situation D or E. When they do fit, the remaining memory is used to hold $N_{LogSwap}$ logically swapped users, with

$$N_{LogSwap} = \frac{MemorySize}{WS_{normal}} - N_{active}$$

Once the number of logically swapped users is known, a new estimate of swap probability can be obtained. As a first approximation, it is assumed that the logical swap probability is proportional with the number of logically swapped users. The physical swap probability then equals [Lazowska 1984]

$$P_{swap} = 1 - \frac{MemorySize/WS_{normal} - N_{active}}{N_{terminal} - N_{active}}$$

The numerator gives the number of logically swapped users, the denominator gives the number of users not currently active. In a real system, the swap probability would probably be less than the value estimated here, since the most active users tend to remain in memory (SRM based its swap decision on the previous think time). These users also generate the largest part of the load.

The P_{swap} found from the equation above is then used to improve upon the first estimate for the swap service demand, which set the probability P_{swap} to 1. The model is solved again for the new P_{swap}, and a new N_{active} results. The process is repeated until P_{swap} is constant.

3.4 *Heavily loaded system (situation D)*

In situation D all active users are kept in memory, but their working set must be reduced below WS_{normal}. SRM cannot control the memory utilization through the logical swap mechanism, since all users are swapped for each transaction. Instead, SRM will initiate page stealing, thus reducing the average working set for the active tasks, and making room for more tasks.
The system is in situation D if

$$MemorySize/WS_{normal} < N_{D,active} < MemorySize/WS_{min}$$

Since the number of active users $N_{D,active}$ is proportional to both the response time and the throughput (Little's Law), a reduction of the working set size from WS_{normal} to WS_{min} may not have the desired effect of making room for more users: the reduced working set results in higher paging rates (D_{page} increases), increased response times and ultimately in an increased time-space product (a large page-seconds value in the MVS terminology).

In solving the model, the working set size is reduced in a few steps. At each step, the paging demand is computed using $P_{pagefault} = f(WS)$. The system submodel is used to compute the number of users $N_{D,active}$ in the system. If it is in the proper range, a better estimate is obtained through iteration. If no $N_{D,active}$ can be found for any UIC, the system is in situation E.

3.5 *Overloaded system (situation E)*

In situation E the memory cannot hold all active users; exchange swapping is used by SRM to limit the number of swapped in users. Since the UIC is kept constant at its minimum value, the number of page faults per transaction will be constant (but high). The page fault probability equals $P_{pagefault} = f(g(UIC_{min}))$. Each task in memory is kept at WS_{min}; the number of swapped in users equals

$$N_{swappedin} = MemorySize/WS_{min}$$

In the model for exchange swapping both the I/O overhead for each exchange swap and the wait time while swapped out must be included.

We first solve a system submodel with $N_{swappedin}$ customers, assuming a single swap in and swap out for each transaction. This submodel yields the maximum throughput the system can handle, at the given swap probability. Note that this submodel differs from the submodel discussed previously in that no terminals are included. Instead, the throughput from the submodel without terminals is used in a two server model (system and terminals) to determine the average number of waiting users. The waiting users in the model are equivalent to swapped-out users; they are "out ready" in terms of the SRM.

The number of swaps per transaction is now approximately

$$P_{swap} = 1 + \frac{2N_{outready}}{N_{active}} \cdot \frac{CPUload}{ISV}$$

and

$$N_{outready} = N_{active} - N_{swappedin}$$

with

CPUload the average service demand for a transaction

ISV the CPU service provided to a customer in a single swapped in interval.

The term "1" in the equation says that each job is swapped in when the transaction starts. The factor $CPUload/ISV$ says that a job is exchange swapped after it used *ISV* service[6] from the processor.

6. The ISV parameter corresponds to the SRM ISV parameter on the performance group command in the IEAIPS PARMLIB member.

The factor $N_{outready}/N_{active}$ says that the number of swaps is proportional with the fraction of users swapped out. The factor 2 says that an exchange swap involves the movement of two jobs. Note that the N_{active} includes both the "in ready" ($N_{swappedin}$) and the "out ready" users.

After the first evaluation of the system submodel a new estimate for the number of swaps per transaction can be obtained, and the submodel can be evaluated again using a more accurate swap I/O load. The process is repeated until the swap probability is essentially constant.

4. Model results and discussion

To use the model we need to make estimates for quite a few system parameters. The system modeled consists of a processor, eight user 3380 disks, eight paging disks (3380 or solid state), and sixteen 3380 swapping disks. As described before, channel and controller contention are modeled in the I/O subsystem. A single time sharing workload is assumed, with a varying number of terminals.

The load and system parameters used are specified in Table 1, in terms of instructions (for the processor) and I/O requests (for the disks).

TABLE 1. Workload and system description.

workload:		
CPU load	500.000	instructions
I/O load	5	I/O
system:		
processor speed	18	MIPS
page fault CPU overhead	10000	instructions
page fault I/O load	1	I/O
swap CPU overhead	25000	instructions
swap I/O load	3	I/O
pages per swap	12	pages
Memory Size	32 or 64	MByte

Furthermore the working set and paging characteristics used are given in Figure 4 as a function of the UIC. The working set size is expressed in Kbytes, the page fault probability in the number of page faults per 1 million instructions.

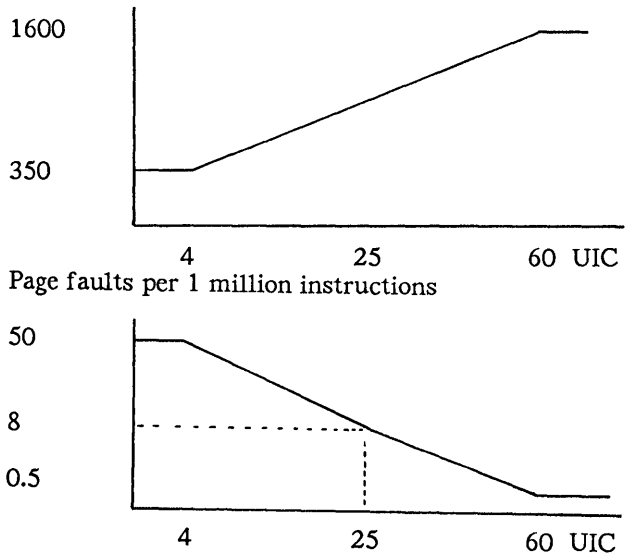


Figure 4. Working set size and number of page faults per 1 million instructions as a function of the UIC.

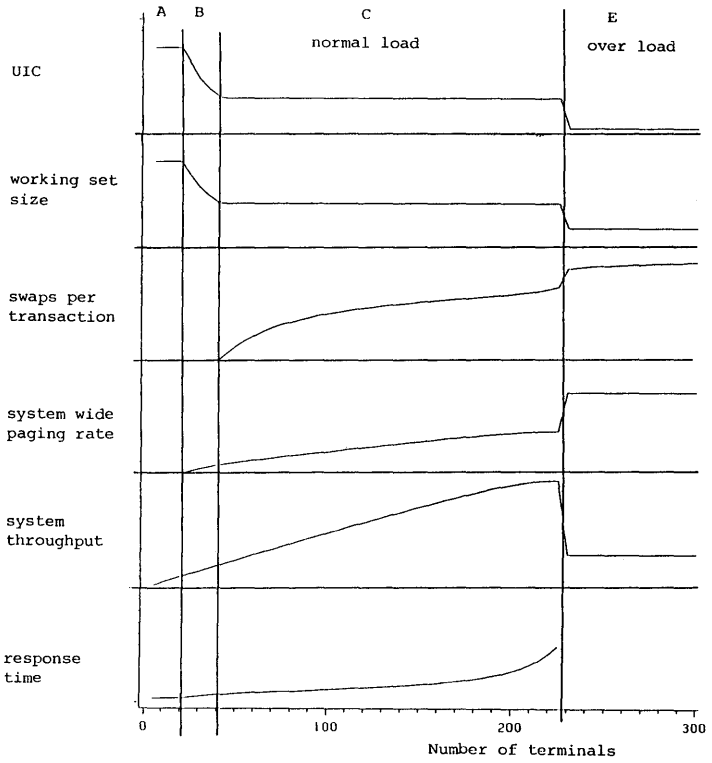


Figure 5. Model results for a system with 32 Mbyte memory and 3380 paging disks.

Figure 5 shows the modeling results for a system with 32 Mbyte of real memory, using standard 3380 disks. The system configuration and workload are dimensioned to create a CPU and/or paging bottleneck; ample capacity is provided for swapping and user I/O.

The system shows a reasonable behavior in the situations A to C, but it obviously cannot handle the paging load in the overloaded system situation (E): the throughput suddenly drops. In the normal load situation C the paging demand is constant for individual transactions, but the system wide paging rate increases significantly, proportional with the system throughput. The number of swaps per transaction increases from 0 to 1 in situation C. It is interesting to note that the heavy load situation (D) does not show up in these results: apparently the working set reduction was offset by the large increase in paging. The sudden collapse in performance in this system is due to a paging bottleneck.

In a real MVS system under SRM control this would not happen if the CPU utilization or paging rate exceeded one of the thresholds, explained before.

The response time has also been evaluated for three other configurations, with upgraded main memory (64 instead of 32 Mbyte) and/or faster paging I/O (solid state disks instead of 3380). The results are shown in Figure 6. The systems shown are

1. 32 MByte main memory, 3380 paging disks,
2. 32 MByte main memory, solid state paging disks,
3. 64 MByte main memory, 3380 paging disks,
4. 64 MByte main memory, solid state paging disks.

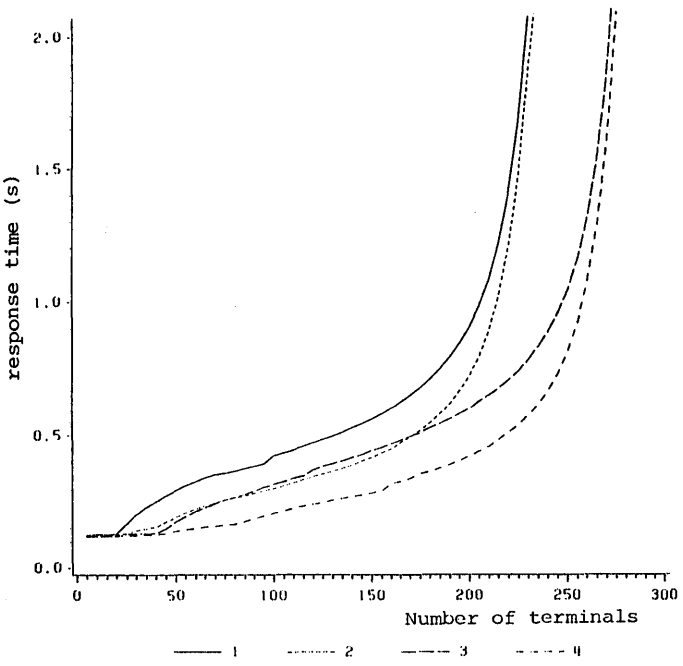


Figure 6. Model results for four different memory and paging configurations.

Obviously, the system with the smallest main memory and the slowest paging disks performed worst. It is interesting to compare the systems 2 and 3: the system with the fast paging disks (2) performs similar to the system with a large memory (3) for up to 200 terminals. In terms of a large MVS system this means that a fast paging device (solid state disk or cached paging controller), which provides very fast access to page datasets, can be a viable alternative for a memory upgrade. At higher loads (more than 200 terminals), the memory constraint really shows up. At high paging loads the CPU overhead and channel bandwidth limit the system performance; a fast paging device no longer provides a performance advantage. Therefore, the response time for the DASD and solid state systems converge.

Of course, all results depend very much on the assumed relation between UIC, working set size, page fault probability and configuration.

5. Model extensions

The modeling approach is based on the fixed relationships between the UIC, working set size and the page fault probability. For a multi-class model the same reasoning can be applied: the working set size and the page fault probability can be computed for each class independently from the UIC, which is of course the same for all classes.

Equations that involve the working set size must now be written as vectors, e.g. the system is in the lightly loaded situation B if and only if

$$\sum_t N_{t,t} WS_{max,t} > MemorySize > \sum_t N_{t,t} WS_{normal,t}$$

Similar reasoning applies to the other situations, although the formulas for the normal and overload situations are somewhat more complex. In these situations it is crucial to define how the classes compete with each other: should they receive the same treatment (i.e., the same swap probability) or should preference be given to a certain class. Basically, both assumptions can be handled with appropriate formulas.

The two level modelling approach, with a high level model describing the SRM algorithms and a low level model describing the system's hardware can also be applied to model other facets of SRM behaviour. For example, the MPL controls which are based on the system wide paging rate and CPU utilization could be included in the model. Also, a more detailed model could be used for paging and swapping reflecting some of the control mechanisms used by the Auxiliary Storage Manager (ASM).

Such extensions are feasable, since the two level model can use the same feedback control as the operating system components.

6. Measurement and validation procedures

Until now no reference was made to any measurement data. Still, for the model to be useful it is crucial that the relationships between the UIC, working set size and page fault probability are known.

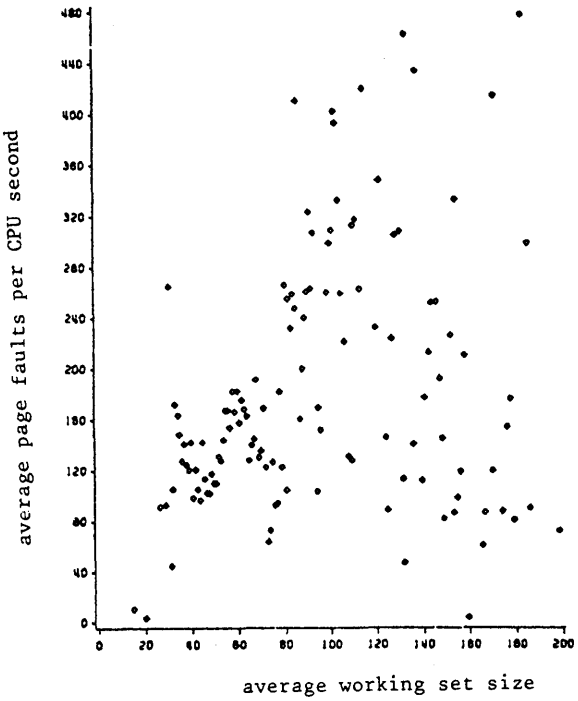


Figure 7. Average number of page faults per (3083) CPU second used, as a function of the average working set size for TSO ISPF users.

In the IBM MVS system, logging data is available on the average working set size, the number of page faults and the CPU time used for each completed task. This data seems particularly useful for obtaining the relation between the average working set size and the paging rate for a given workload. Unfortunately, this does not work very well in practice: in a "real" system the workload tends to be too heterogeneous, i.e. the number of

page faults per CPU second plotted against the average working set size tends to show little correlation.

Figure 7 shows the average number of page faults per CPU second as a function of the average working set size for all TSO-ISPF users in the system. Each point represents the average for a specific number of pages. This plot does not show a reduction in the number of page faults as the working set increases. Instead, it seems to tell that larger programs generate more page faults. The problem with such measurements is that the individual observations represent different users, running different programs. This makes it very difficult to obtain useful model parameters.

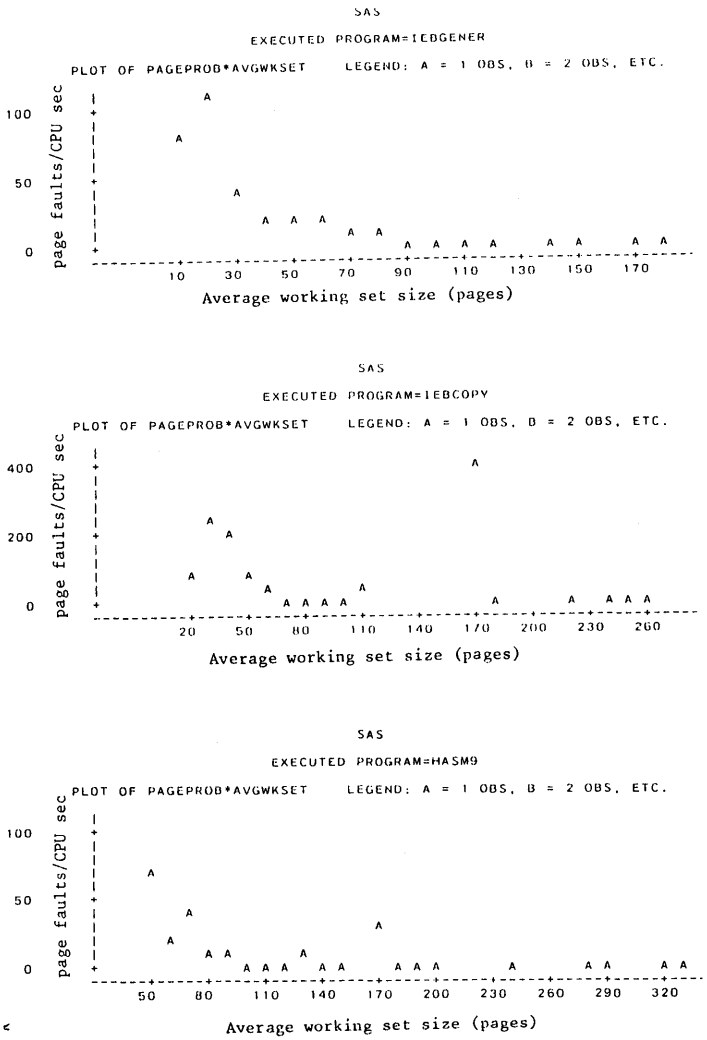

SAS
EXECUTED PROGRAM=IEBGENER
PLOT OF PAGEPROB*AVGWKSET LEGEND: A = 1 OBS, B = 2 OBS, ETC.
page faults/CPU sec
100
50
0
10 30 50 70 90 110 130 150 170
Average working set size (pages)
SAS
EXECUTED PROGRAM=IEBCOPY
PLOT OF PAGEPROB*AVGWKSET LEGEND: A = 1 OBS, B = 2 OBS, ETC.
page faults/CPU sec
400
200
0
20 50 80 110 140 170 200 230 260
Average working set size (pages)
SAS
EXECUTED PROGRAM=HASM9
PLOT OF PAGEPROB*AVGWKSET LEGEND: A = 1 OBS, B = 2 OBS, ETC.
page faults/CPU sec
100
50
0
50 80 110 140 170 200 230 260 290 320
Average working set size (pages)

Figure 8. Average number of page faults per (3083) CPU seconds used, as a function of the average working set size, for the programs IEBGENER, IEBCOPY and HASM9.

Better results can be achieved by monitoring a single, known job, as reported by [Bonnes 1986]. The analysis of results for the IEBGENER, IEBCOPY and HASM9 (assembler) programs are shown in Figure 8. They show the expected behavior: the number of page faults per CPU second increases as the average working set decreases. Unfortunately most of the programs do not show such "nice", predictably, behavior.

For a single class model it would be sufficient to know the relation between paging and working set size, as plotted in the Figures 7 and 8. For a multi class model we must know the UIC since it is used to link the working set size of the classes. The UIC, however, is not measured by the standard monitoring tools, so no long term measurement data is available. RMF trace records [IBM 1982] could be used to collect UIC information for specific time periods.

The only acceptable method to *validate* the model presented here is to create a benchmark (test) workload, using a terminal simulator. With such a system, it is possible to perform detailed measurements on a known workload, with a varying number of terminals. This is the method used by hardware manufacturers to compare their systems. Unfortunately, this method requires two dedicated MVS systems to be available for extended periods of time: one to generate the workload, and one to process it.

7. Conclusions

The model as described allows for a reasonably detailed model of the SRM memory control algorithms. In its current form, the model can be used for simple studies on the relations between paging and the memory size of a system. It is easily possible to extend the model to the multi class case, where each class can have its own characteristics. Also, because the model is based on iterative schemes it seems possible to include more of the SRM control mechanisms, notably the domain and performance group concepts. The main problem with the approach is that the parameters, although defined in rather elementary quantities, are hard to obtain from a real workload. We are currently investigating various methods to derive the parameters used from measurement data.

8. References

[Bonnes 1986] A.H.J. Bonnes, **Measuring Memory usage in MVS systems,** presentation at ECOMA-14, Brussels, 1986.

[Beretvas 1986] T. Beretvas, **The evolution of paging algorithms in VM and MVS,** ECOMA-14 proceedings, Brussels, 1986, pp 271-288.

[Denning 1980] P.J. Denning, **Working sets past and present,** IEEE transactions on Software Engineering, Vol 6, No 1, January 1980, pp 64-84.

[IBM 1982] **RMF Field description and Analysis, Version 1, MVS/SP 1.3.0,** IBM manual GG24-1564-00, June 1982

[IBM 1985] **MVS/XA SPL: Initialization and Tuning Guide** IBM manual GC28-1149, 1985.

[Lazowska 1984] E.D. Lazowska, J. Zahorjan, G.S. Graham and K.C. Sevcik, **Quantitative System Performance, Computer System Analysis Using Queueing Network Models,** Prentice Hall, Englewood Cliffs, 1984.

[van Someren 1984] J. van Someren, **Paging in Berkeley UNIX,** report 051560-44(1984)01 Delft University of Technology, 1984.

PROFS Capacity Planning Issues

Allan H. Shub
Duke Power Company

ABSTRACT

Capacity Planning in the VM/PROFS environment is, in some respects, less complicated than MVS capacity planning. It is not, however, an exact science, nor can it be applied without an understanding of certain relevant issues and techniques. This paper presents a discussion of some of the more important issues a PROFS capacity planner must address. One of the most significant issues for large PROFS systems is the real storage constraint and the associated limitations imposed on the ability to exploit the capacity of large processors (e.g., IBM 3090-200). Other issues discussed include DW/370 vs. DCF resource requirements and alternative PROFS configurations.

This discussion of PROFS (Professional Office System) capacity planning is not intended to be a comprehensive guide or tutorial. The intent is to highlight several important issues that many PROFS capacity planners must address. Among the issues considered are the relative resource requirements of DW/370 (Displaywrite/370) and DCF (Document Composition Facility), the real storage constraint associated with large PROFS systems and the paucity of data sources and analytical aids.

DATA SOURCES AND ANALYTICAL AIDS

A significant problem facing many PROFS capacity planners and analysts is the acquisition of useful and appropriate resource utilization data. The PROFS capacity planner has relatively simple, homogenous workloads to analyze. It is a relatively simple task to profile the resource requirements for the 'typical' PROFS system. There are relatively few subsystems (e.g., VTAM) and several service machines providing reasonably well defined and limited services (e.g., mail and calendaring). Contrast this to the nightmarish complexity of large MVS/XA online transaction processing systems requiring CICS (with ISC or MRO), IMS or DB2, and several acres of large processors and peripheral equipment. That is the good news.

The bad news is that even though VM systems are generally much simpler than MVS systems, performance monitors, analytical aids and planning tools in the VM environment are severely limited in comparison to the extensive choice available to the MVS capacity planner. VM/Monitor records, for example, are a mere wisp of data compared to SMF/RMF records. Beyond the fact that there are insufficient data, some of the data that are collected are of questionable value. VM/Monitor contains some wonderful anachronisms. Outbound message lengths, for example, are not reported beyond 128 bytes. This is an artifact of TTY terminal protocols. Internal host response times are often estimates rather than measured values. VMMAP (a data analysis facility for VM/Monitor records) and VMPPF (an analytic modeling system also using VM/Monitor data) are supported by different IBM groups in Cary, NC. Both products estimate rather than measure response time. VMMAP and VMPPF employ different algorithms and often yield widely divergent results. (And, of course, each product group contend their estimates are correct.)

As a result of inadequate monitoring facilities, seemingly simple problems can prove to be exceedingly challenging. Suppose, for example, your management requests a functional profile of the 'typical' PROFS user (i.e., how many times per hour or day does this person use notes, calendaring, or document preparation and what are the resource requirements of these functions?). VM/Monitor records contain no useful information with respect to this analysis. The PROFS Programmers Guide for PROFS Release 2.2 (TNL SN20-4668-0) documents accounting exits which may be used to capture the necessary data. The time and personnel resources necessary to code and implement these exits are considerable. Worse yet is the performance caution on the last page, "you risk degrading performance . . . all users will cause your exit to be executed many, many times in the course of a day." Further, so much data can be generated that you may overflow the spool areas. Poorly designed exits, improperly used, could cripple the system. There are third party performance monitors available which will monitor at the exec or module level quite competently (e.g., CMAP). Unfortunately, PROFS is designed with numerous functions incorporated into one module. To circumvent this problem, it is necessary to replace main menu commands with dummy execs, which in turn invoke the PROFS module. Frequency distributions can be generated directly from this data. Resource utilization analysis requires additional post processing (e.g., SAS) to examine the invocations of the PROFS module immediately following the execution of a dummy command.

Of what use is such an analysis? Why jump through all these hoops? If, for example, you are considering replacing DCF with DW/370 it is very important to understand what proportion of the workload DCF activity constitutes. If DCF activity is relatively limited, substitution of DW/370, even if it requires significantly more resources, may be inconsequential and may be implemented with abandon. If, on the other hand, DCF is a major constituent of the workload, a DW/370 substitution may merit a more cautious implementation.

A brief statement concerning performance modeling software: this is one of the few areas where VM support approaches MVS levels. There are two major participants in this area: IBM and BGS Systems. IBM's entry, VMPPF must be recognized because it is from IBM. It is a competent product in most respects and is very easy to use. IBM has a significant advantage in developing support for new products quickly: it has ESP and other interval data to which competitors do not have access. The most significant limitation of VMPPF is the inability to model storage utilization properly, especially with respect to the real storage constraint problem discussed later in this paper. Further, it is not clear what role VMPPF will have in the future. IBM will not disclose how many installations have acquired VMPPF, but it does not appear to have sold well. Enhancements to VMPPF have consisted primarily of support for new products. VMPPF has demonstrated little functional evolution.

BGS System's BEST-1/VM, on the other hand, is evolving rapidly with respect to functional capabilities. Capacity planning software is the lifeblood of BGS Systems and their commitment to support their products is strong and convincing. BEST-1/VM cannot model the real storage constraint, and unlike VMPPF, it cannot effectively model the relationship between PROFS users and the various PROFS service machines. BGS Systems is considering the real storage constraint problem (lack of data may

prohibit resolution). Modeling the linkage between PROFS users and the service machines is expected to be supported by mid-1987.

DW/370 VS. DCF

One of the more interesting PROFS capacity analyses recently conducted at Duke Power Company was the comparative analysis of DW/370 and DCF. The most interesting aspect of this study is that the conclusions differ significantly from recently distributed IBM SNAPSHOT studies (IWSYS Flash F8644, Professional Office System (PROFS) Capacity Planning, 11/28/86). The IBM study indicates that DW/370 document preparation requires approximately 15% less processor capacity than DCF document preparation. The Duke Power study demonstrates that DW/370 requires significantly more processor capacity than DCF (in some situations). More significantly, the Duke Power study also revealed that DW/370 generates significantly more I/O activity than DCF. Total non-paging I/O was 50% to 165% higher and page reads were measurably higher. The results of the Duke Power study include:

- The trivial/non-trivial ratio was consistently between 4-5:1 for the entry/update of the one page documents. However, this same measure is between 1.2-1.7:1 for the entry/update of the five page document with DW/370. The reason for this behavior is the 'TSPELL' function of DW/370.

- The total number of transactions with DW/370 was 17-54% higher than that generated by DCF.

- The total cpu time of DCF was 0-14% higher than that of DW/370 for short documents. For long documents, DW/370 consumed 19-100% more cpu time (100% for document entry, and 19% for document change).

- The total non-paging I/O of DW/370 was 46-165% higher than that of DCF (the higher increases, 91% and 165%, were for the update and entry of the long documents). The primary contributor to this increase appears to be the 'TSPELL' feature of DW/370. Comparing spelling error correction with TSPELL to PROOFREAD (DCF) shows that TSPELL generates five times more I/O activity and three times more processing time.

- The average working set size required for DW/370 was 6-9% higher than that required for DCF for the long document. For short documents, DCF required an average working set size 5-14% greater than that required by DW/370.

- The average number of page reads per transaction observed for DW/370 was 2-15% higher than that observed for DCF for the long document. For short documents, the average number of page reads per transaction observed for DCF was 26-130% higher than for DW/370.

- Although the calculated trivial transaction response time for DW/370 documents was 24-74% higher than that calculated for DCF, the calculated non-trivial transaction response time for DW/370 documents was 54-157% lower than that calculated for DCF. The net effect of this phenomenon is that DW/370 appeared to have more consistent response times.

Refer to Tables 1 and 2 and Figures 1 through 4 for detailed results.

Actual impact of DW/370 on the system will depend on several factors: the capacity of the processor, amount of real storage, the capacity of the I/O subsystem, and the proportion of the workload that will be converted to DW/370. If the system under consideration has an I/O subsystem that is heavily loaded, conversion to DW/370 may entail some risk.

Why do two studies examining the same products produce divergent impressions of the relative resource requirements of DW/370 and DCF? Both studies employed many of the same tools and facilities. The difference in conclusions illustrate the importance of the scope of the study and of initial assumptions, conditions, and limitations on the outcome of the studies.

In their study, IBM adopts a rather limited view of capacity planning and examines only processor utilization. It is assumed that there is no paging, I/O or memory contention. This limited approach is consistent with IBM's USAGE method of capacity planning (VM-USAGE Capacity Planning, GG22-9045-00). It is a useful assumption for comparing the capacity of diverse processors, but it is debatable that it is a useful capacity planning methodology. The higher I/O rates associated with DW/370 illustrate the problem with this assumption. The balanced systems methodology advocated by some within IBM is a better methodology for this problem (Blanced Systems and Capacity Planning, GG22-9299-00).

Both the IBM and Duke Power studies used Teleprocessing Network Simulator (TPNS) to simulate user activity. (TPNS is a testing package that facilitates the evaluation of network and host systems, application software and operating systems without the actual use of a terminal. TPNS uses scripts to generate transactions. While developing these scripts can be a labor intensive effort, it is much less expensive to have one person invest several days developing scripts, than it is to have tens or hundreds of people entering documents from terminals in a coordinated manner for an hour or two. TPNS tests are easily repeatable, and greatly reduce the possibility of human error.)

Perhaps the most significant difference between the two studies is the scripts employed to examine the differences between DW/370 and DCF. The IBM script used only one document. This document had five paragraphs with a total of twenty-three lines. The Duke Power script included entering and updating three different documents of varying lengths (one to five pages) and differing formats. The Duke Power script attempted to minimize the use of the 'ENTER' key. After the completion of the document it was proofread (the same errors were entered in each case), and all errors were corrected. The document was then viewed and returned to personal storage.

Both analyses used PROFS 2.2. IBM collected data from an IBM 4341-P12 with 16 megabytes of real storage operating under VM/SP Release 3.0. Duke Power collected data from an IBM 4381 P-02 with 16 megabytes of real storage operating under VM/SP HPO 4.2.

The differences in the two comparisons of DW/370 and DCF certainly lend credence to the caveat that prefaces most IBM performance and capacity analyses, roughly paraphrased, "these results are reliable only for these conditions and these specific workloads". While we believe that the SNAPSHOT results are reasonable for the workload specified, that workload, by itself, is not a reasonable representation of the workloads found at Duke Power. This leads us to conclude that the Duke Power study may be more representative of the differences between DCF and DW/370 and certainly is more appropriate for our installation. Do your own capacity planning. Do not depend on a vendor or an analyst at another installation to understand your workloads.

REAL STORAGE CONSTRAINT

For very large PROFS installations (1000 or more logged on users on a single processor, or CEC) a new and troublesome problem recently has been recognized: The Real Storage Constraint. The real storage constraint may make the exploitation of an IBM 3090-200 class processor difficult and perhaps impossible under certain circumstances. VM/SP HPO implements 24-bit addressing archi-

tecture which permits addressing up to 16 megabytes of real storage. The CP portion of HPO is implemented without dynamic address translation (DAT), and therefore restricted to addressing the real storage below the 16 megabyte line. Areas referenced by CP must also reside below the 16 megabyte line. These areas include trace tables, control blocks (FREE STORAGE) and user pages (DPA).

Free storage is not pageable. Each virtual machine in the system requires as much as three pages of free storage. Free storage contains the page and swap tables as well as other control blocks. The VM/SP HPO Planning Guide and Reference (SC19-6223-6) states that a modest virtual machine configuration requires less than one page of free storage. Based on this estimate, 1200 PROFS users would require about five megabytes of free storage. Unfortunately, the IBM estimate is unrealistic. Two or three pages is more realistic (at Duke Power, it requires approximately 2.6 pages of free storage per virtual machine). At approximately three pages per machine, 1200 machines require 3600 of the 4000 pages below the 16 megabyte line. (Figure 5 illustrates the usage of real storage below the 16 megabyte line.) Remove another 400 of 500K for the nucleus, and not much is left for user pages (DPA). The enormous fixed storage requirements of large VM systems effectively limits the exploitation of large IBM 3090 processors with respect to workloads that have a large number of users (e.g., PROFS). Much of the power of an IBM 3090-200 may be wasted if large numbers of users cannot be logged on because of the real storage constraint. An IBM 3090-200 has the processor capacity to support 2000 or more active PROFS virtual machines, but until the real storage constraint is resolved, it may be limited to 1200 or so logged on.

The ultimate solution is, obviously, to implement 31-bit addressing. VM/XA is the VM operating system that supports 31 bit addressing. VM/XA's control program can address two gigabytes, and the real storage constraint simply does not occur. The current implementation of VM/XA (VM/XA System Facility) is a limited operating system; or as IBM so delicately states, it "lacks some of the functional richness of HPO." That means that it does not support CMS (or PROFS) among other things. To use VM/XA SF as a solution requires VM guests and the performance trade off associated with guest machines (90% of native performance for the preferred guest and leftovers for the other guests). PROFS communities may may also have to be split to separate VM guest machines (the real storage constraint applies to the guest systems). General availability of full function VM/XA SP is not until March, 1988. Release 1 will not support native VTAM; that support will be in Release 2 (1Q 1989).

Assuming the VM/XA SF solution is not appealing and the problem must be addressed before the availability of VM/XA SP, there are a variety of options that will provide limited relief for the real storage constraint. Managing VM/SP HPO Real Storage Constraint Below the 16 Megabyte Line, (GG66-0261-00) documents 31 actions that may be taken to relieve the real storage constraint. (Also refer to VM/HPO Performance Tuning Guide (GG22-9392-01) for additional tuning suggestions). The single, most effective action is the installation of VM/SP HPO 4.2. HPO 4.2 is much more aggressive in migrating page and swap tables of inactive users. The only criteria for migration in HPO 4.2 is that page and swap tables comprise at least 12% of unextended free storage, and it is easily met. At Duke Power, implementation of HPO 4.2 reduced free storage requirements by approximately 2.5 megabytes for 850 logged on users. VM/HPO 5.0 will further reduce free storage requirements.

The PROFS environment can be changed in several ways to better exploit the improved migration of page and swap tables offered by HPO 4.2. (These changes may reduce the function or convenience of PROFS and may be unpalatable to portions of the PROFS user community.) These changes include:

- increase the PROFS timer interval as much as possible. Without modifying PROFS, the maximum value is 999 seconds (or 16.7 minutes). The only function affected by this change is REMINDERS. The benefits of increasing the timer interval are significant. In addition to increasing the number of users considered inactive and eligible for migration, significant reductions in paging and swapping may be realized. At Duke Power, the interval was changed from one minute to five minutes. Swapping and paging rates decreased by one-third and one-fourth, respectively.

- in later releases of PROFS, the interval timer may be selectively disabled. To minimize page/swap activity and free storage requirements, disable the timer for all machines except those having an OFS $remind file at logon and enable the clock for any user who envokes the reminder function.

- implmentation forced logoff. Timeout will release additional free storage. Even if an inactive virtual machine's page and swap tables have been migrated, approximately one page of free storage remains resident in real storage. At Duke Power, 850 to 900 users are usually logged-on at any given time. Analysis of VM/Monitor data indicates that only 350 to 400 (approximately 40%) of the logged on users are active during a 15 minute period (i.e., performing some task other than the servicing of timer pops. Refer to Figure 6). Forcing off those machines that are inactive would release an additional 1.5 to 2 megabytes of free storage. The timer interval must be longer than the timeout interval, otherwise no virtual machines will be timed out.

- circumvention or resolution of the real storage constraint may only provide limited expansion capabilities. There is reason to believe that VTAM in the VM environment can support only about 1400 concurrent users. Detail of this constraint and possible plans for resolutions are not generally available.

PROFS capacity planning is more of an art than a science. Improvements in products such as BEST-1/VM and the availability of more monitoring facilities offer hope that this situation will change in the not too distant future. It cannot be overemphasized that it is imperative that performance expectation and forecasts must be based on workloads and conditions similar to those which are anticipated.

TABLE 1

ENTER LONG DOCUMENT

Ratio of trivial to non-trivial transactions- -	DCF	is 5.5:1	
Ratio of trivial to non-trivial transactions- -	DW/370	is 1.2:1	
Total number of transactions- - - - - - - - - -	DW/370	is 17%	higher
Total number of trivial transactions- - - - - -	DCF	is 33%	higher
Total number of non-trivial transactions- - - -	DW/370	is 247%	higher
Total virtual time- - - - - - - - - - - - - - -	DW/370	is 106%	higher
Total virtual time for trivial transactions - -	DW/370	is 12%	higher
Total virtual time for non-trivial transactions	DW/370	is 120%	higher
Total cp time - - - - - - - - - - - - - - - - -	DW/370	is 84%	higher
Total cp time for trivial transactions- - - - -	DCF	is 14%	higher
Total cp time for non-trivial transactions- - -	DW/370	is 157%	higher
Total cpu time- - - - - - - - - - - - - - - - -	DW/370	is 100%	higher
Total non-paging I/O- - - - - - - - - - - - - -	DW/370	is 165%	higher
Total trivial non-paging I/O- - - - - - - - - -	DW/370	is 110%	higher
Total non-trivial non-paging I/O- - - - - - - -	DW/370	is 177%	higher
Working Set Size- - - - - - - - - - - - - - - -	DW/370	is 6%	higher
Total page reads(pg/sw) for all transactions- -	DCF	is 13%	higher
Total page reads for trivial transactions - - -	DCF	is 104%	higher
Total page reads for non-trivial transactions -	DCF	is 110%	higher
Response time(all Transactions) - - - - - - - -	DW/370	is 68%	higher
Response time(trivial transactions only)- - - -	DW/370	is 24%	higher
Response time(non-trivial transactions only)- -	DCF	is 54%	higher
Total cpu time(virtual/cp) per transaction- - -	DW/370	is 70%	higher
Total non-paging I/O per transaction- - - - - -	DW/370	is 126%	higher
Total page reads(pg/sw) per transaction - - - -	DW/370	is 15%	higher

TABLE 2

UPDATE LONG DOCUMENT

Ratio of trivial to non-trivial transactions- -	DCF	is 5.5:1	
Ratio of trivial to non-trivial transactions- -	DW/370	is 1.7:1	
Total number of transactions- - - - - - - - - -	DW/370	is 26%	higher
Total number of trivial transactions- - - - - -	DCF	is 6%	higher
Total number of non-trivial transactions- - - -	DW/370	is 203%	higher
Total virtual time- - - - - - - - - - - - - - -	DW/370	is 11%	higher
Total virtual time for trivial transactions - -	DW/370	is 178%	higher
Total virtual time for non-trivial transactions	DW/370	is 3%	higher
Total cp time - - - - - - - - - - - - - - - - -	DW/370	is 39%	higher
Total cp time for trivial transactions- - - - -	DW/370	is 26%	higher
Total cp time for non-trivial transactions- - -	DW/370	is 46%	higher
Total cpu time- - - - - - - - - - - - - - - - -	DW/370	is 19%	higher
Total non-paging I/O- - - - - - - - - - - - - -	DW/370	is 91%	higher
Total trivial non-paging I/O- - - - - - - - - -	DW/370	is 686%	higher
Total non-trivial non-paging I/O- - - - - - - -	DW/370	is 54%	higher
Working Set Size- - - - - - - - - - - - - - - -	DW/370	is 9%	higher
Total page reads(pg/sw) for all transactions- -	DW/370	is 29%	higher
Total page reads for trivial transactions - - -	DCF	is 11%	higher
Total page reads for non-trivial transactions -	DW/370	is 75%	higher
Response time(all transactions) - - - - - - - -	DW/370	is 18%	higher
Response time(trivial transactions only)- - - -	DW/370	is 57%	higher
Response time(non-trivial transactions only)- -	DCF	is 102%	higher
Total cpu time(virtual/cp) per transaction- - -	DW/370	is 7%	higher
Total non-paging I/O per transaction- - - - - -	DW/370	is 51%	higher
Total page reads(pg/sw) per transaction - - - -	DW/370	is 2%	higher

CPU TIME

CONSUMED IN PREPARING, PROOFREADING, AND PRINTING ONE DOCUMENT

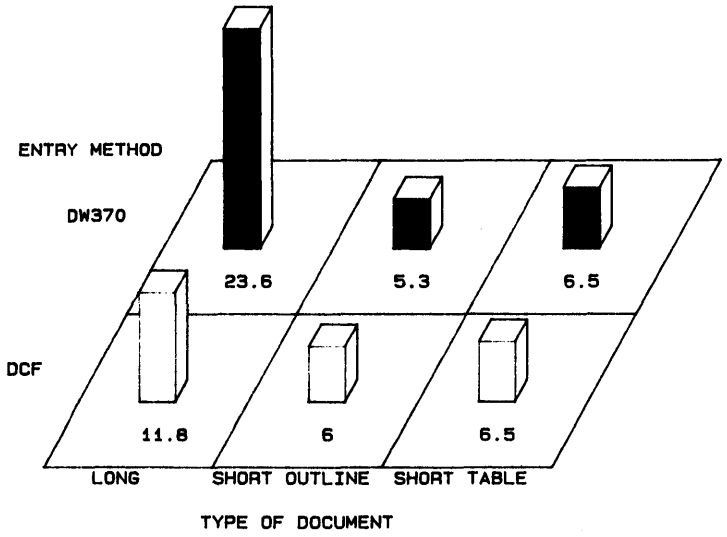


FIGURE 1

CPU TIME

CONSUMED IN PREPARING, PROOFREADING, AND PRINTING ONE FIVE PAGE DOCUMENT

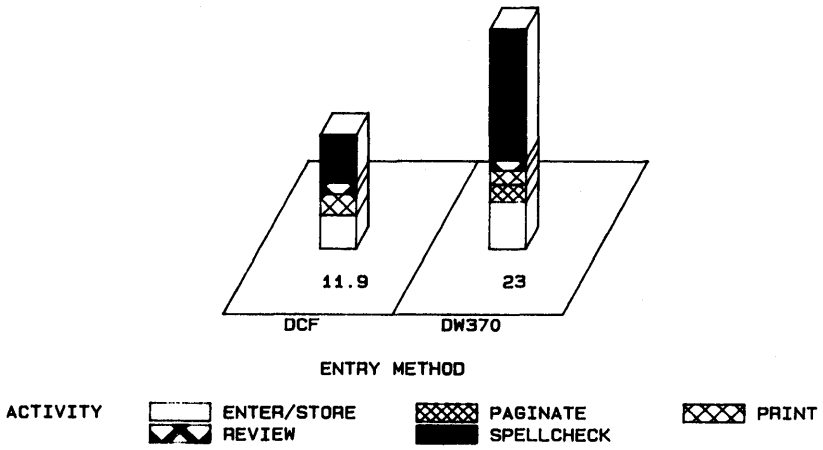


FIGURE 1A

VIO COUNTS

REQUIRED IN PREPARING, PROOFREADING, AND PRINTING ONE FIVE PAGE DOCUMENT

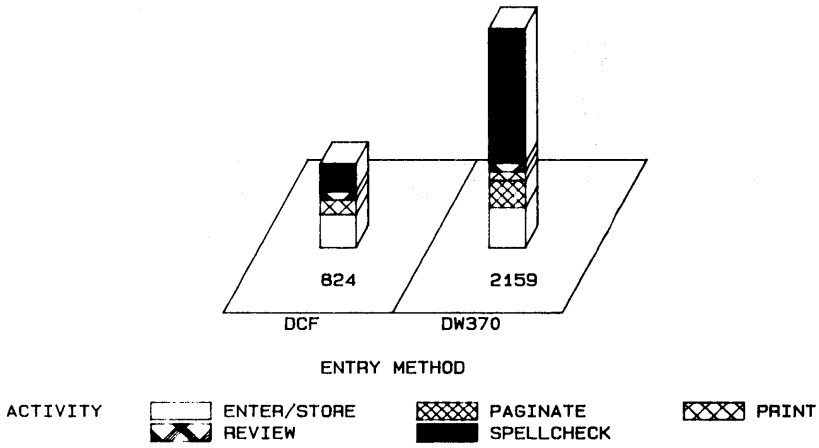


FIGURE 1B

VIO COUNTS

REQUIRED FOR PREPARATION, PROOFREADING, AND PRINTING OF ONE DOCUMENT

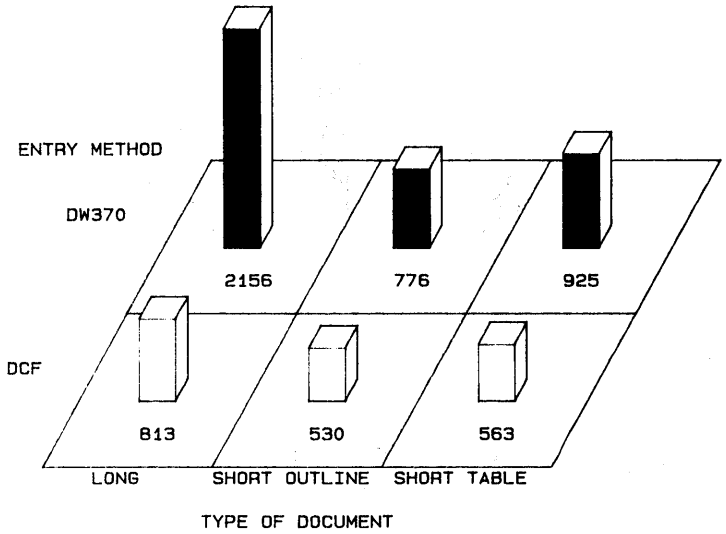


FIGURE 2

PAGE READS

REQUIRED FOR PREPARATION, PROOFREADING, AND PRINTING OF ONE DOCUMENT

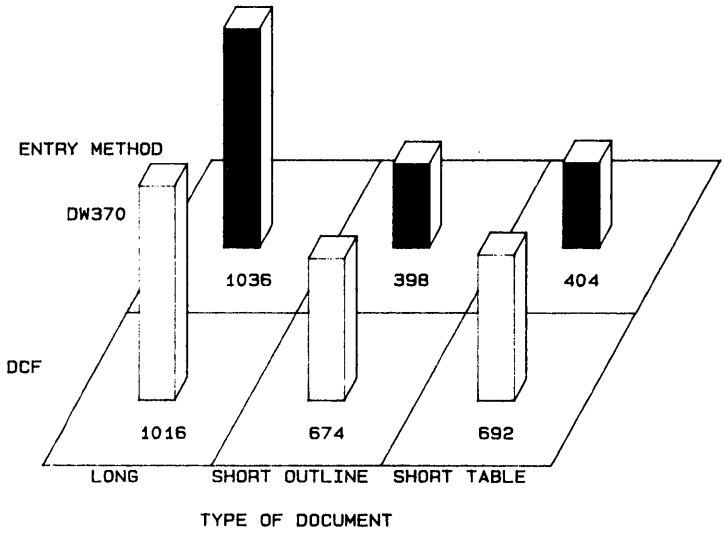


FIGURE 3

CENTRAL STORAGE

REQUIRED FOR PREPARATION, PROOFREADING, AND PRINTING OF ONE DOCUMENT

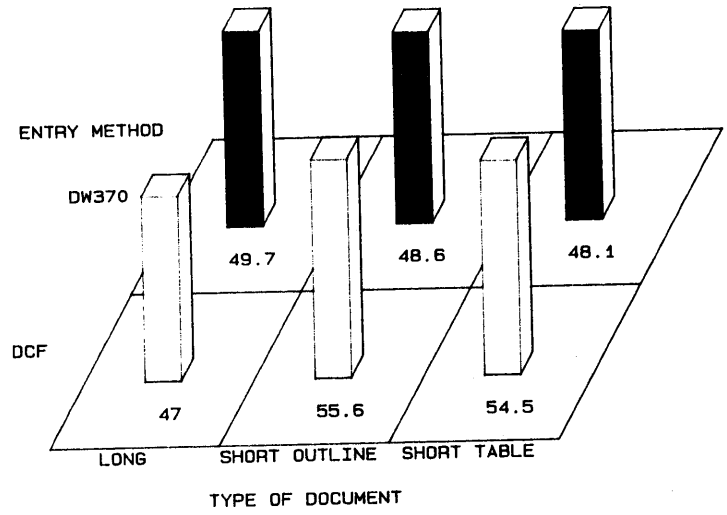


FIGURE 4

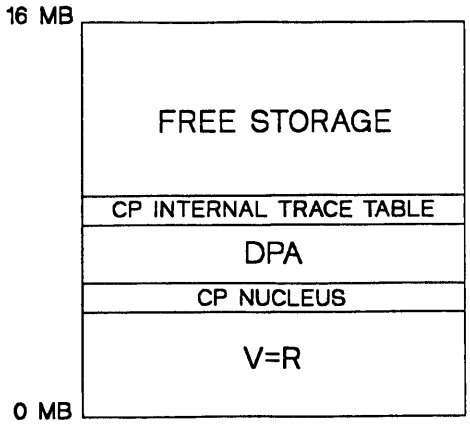
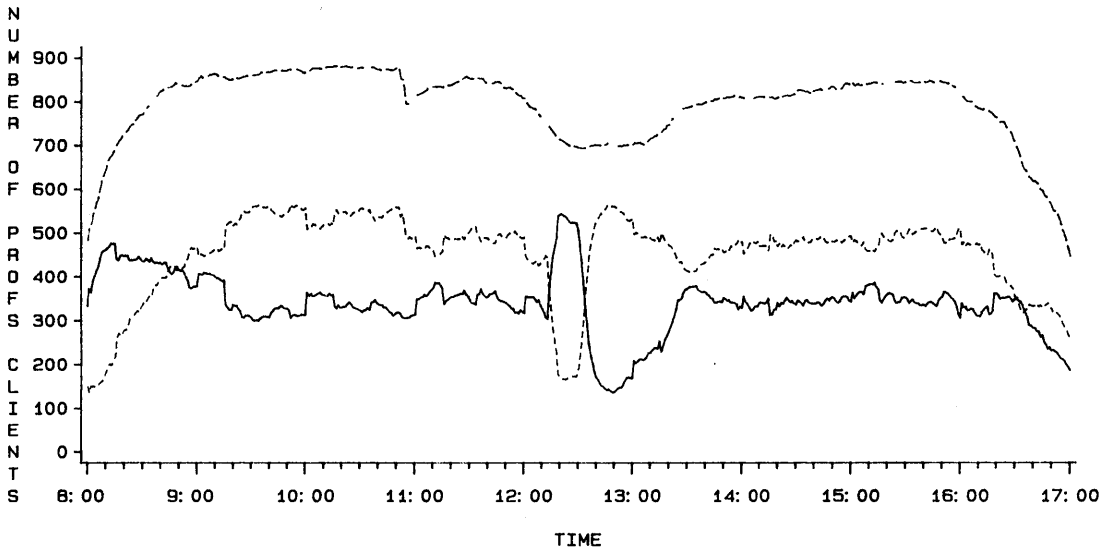


Figure 5

FORCED LOGOFF

SIMULATION OF A 15 MINUTE FORCED LOGOFF FOR THE ENTIRE PROFS COMMUNITY



PLOTS OF ACTIVE, FORCED OFF, AND TOTAL PROFS CLIENTS OVER TIME
FRIDAY FEBRUARY 6, 1987
FIGURE 6

VM PERFORMANCE - TOP DOWN THAT WENT BOTTOM UP

Mary Sue Dickerson
United Airlines

ABSTRACT

The plan was to do hardware planning, bottleneck analysis, and response time profiles for a currently running VM system. This goal was to be accomplished in minimal time and with only 'part' of a person. To achieve this no time could be spent 're-inventing the wheel', so a model was purchased to predict the effects on response time of changing hardware. A data base was developed using the model's output to record changes. But the VM system analyzed was not primarily a CMS machine, and the model would not converge with measured data.

This paper describes some of the top down events that started with a 'sophisticated' VM model but led all the way back down to the 'raw' monitor data. Some of the things learned about VM and the monitor data on the bottom up ascent will be discussed. This paper is specific to VM with the aim of broadening viewpoints for anyone involved in the performance of a VM System.

1. PURPOSE

The purpose for writing this paper is to describe some analysis which can be done easily and without a big expenditure in time or money. At first the monitor data seemed untouchable, but once the initial barrier (partly the sheer volume of data) was broken, a world of opportunity unfolded with little effort. However, since starting from scratch is justifiably not in vogue these days, a background explanation will be given first. This paper will not describe a super in-house developed performance reporting and database system for VM to replace VMMAP, but simply a way to have control over the monitor data for flexibility. This paper will also not describe differences in the various VM systems, but will assume the HPO 3.6 Version of VM.

2. BACKGROUND

A formal capacity planning and measurement effort was made for an existing VM system with about 350 users logged on and 20-25 guest TPF (Transaction Processing Facility) operating systems. TPF is a streamlined operating system which processes on-line transactions quickly. The VM system was used as a development system for TPF and complaints of slow response times were frequent. The system's hardware was primarily back-up for the online system. However, to support these testing requirements, capital expenditures would need to be made. The planning effort was to make sure the money was spent most effectively.

The first step was to purchase VMPPF, a VM model available from IBM. The intent was to validate a model of the current system to obtain a wait/service profile for an existing transaction. The second step would be to alter the system hardware and model its effect on response time. The problem was that the model did not converge due to swapping conditions. In other words, the model was saying our currently running system couldn't possibly be running. But it was.

Disappointment was also experienced by the manual effort to relate the 300 disks (actually more, except monitor did not report on them) to workload classes. More support from the vendor could probably have resolved some of the problems, but this didn't happen. The last recommendation before the model was put on hold was to model CMS with two or three of the TPF systems and block off the resources of the remaining TPF systems used with the 'dedicated batch' facility. The model converged but this meant about 60% or more of the system was blocked off with still no trusted value for transaction response time.

After this experience there was not much support to purchase another product from any vendor. Besides this would have meant a nine to twelve month delay before the software was approved, purchased, and installed. Then training would be required for getting up to speed for the new product. This was not a thrilling prospect while users were complaining daily. The immediate step was to find out how 'bad' a slow response was by being able to enumerate response times.

3. RESPONSE TIME - A MUST

The problem of measuring response times was solved by modifying a service machine which most TPF transactions go through for a CRT interface to the TPF system. The transactions are time-stamped in microseconds as they enter the service machine and again as they leave. These data are then spooled to a reader file, read by SAS, and a response time distribution for each TPF system is saved in a data base. A report by response time distributions was found to be both more consistent and representative than a straight average because a few large response times frequently inflated the average.

The reported response times did not include the trivial CMS response time or any control unit contention (a network was not involved), but the most variable portion of the total time had been captured. Since the virtual user identification was retained in the response data, a user complaining of slow response times could be handed a report listing all his transactions with their respective response times. Handing a user a personal list of response times was dramatic and proved to be valuable in establishing a starting point.

The next task was to identify components which were slowing down the responses and break them up into wait and service components. The methodology used will be termed 'contention analysis' [1]. Although less sophisicated than a model, the method was cheaper in terms of money, people, and time, but yet effective in identifying the VM system capacity constraints. The main drawback is not being able to automatically project response time changes as a result of new hardware. David Potter with the IBM research Laboratory in Yorktown Heights was a great help and many of the following ideas were his.

4. CONTENTION ANALYSIS

The first big breakthrough came when it was learned how to read the monitor records. They are easily read with REXX (VM System Product Interpreter) and the few lines of code to read them have been included in Appendix A. Like a 'kid in a candy store', all the data available create the temptation to re-write VMMAP or spend time and computer resources databasing everything. But the goal was to look at small amounts of data for specific purposes. In other words 'keep it simple' and 'change will be the norm'. One important exception was to put basic system variables such as CPU utilization, paging, and I/O, with the collected response times on a one page management report. SAS was used to read the records created by REXX and to maintain a database.

The contention analysis evaluated various wait states for in-queue users. Most of these values came from the user monitor records (code 400). The wait components were first broken into eligible, in-queue, console function wait, and idle. The in-queue waits are divided into CPU, I/O, page, swap, deferred function, instruction simulation, terminal I/O, run user (not yet dispatched), PSW, and unknown wait. The REXX code to break users into various waits is included in Appendix B. A SAS timeplot then shows these waits for each interval throughout the day.

The percentage of in-queue users in each wait state can be calculated for more specific analysis. SRM tuning parameters can change most of these waits (for example minimum working set size may affect eligibility); however, they will not be addressed further in this paper. Each of the in-queue waits is now examined.

CPU wait (CP) - The workload or hardware needs to be adjusted. A graph of active users by the percentage in CPU wait will give a good indication of the limit for the CPU.

I/O wait (IO) - The monitor device (600-602) records or VMMAP will give an idea of busy devices. Actually this implementation reads the monitor records and lists every occurrence of a device or control unit queue throughout the entire day. Devices or paths with frequent queues are then suspected if I/O wait is significant. I/O contention may occur even if I/O wait percentage is low due to serialization within the application, and once an I/O has been started, the user is no longer considered to be in I/O wait.

SWAP wait (SW) - If swapping wait is high, the most likely solution is to upgrade DASD or rearrange the DASD subsystem.

Deferred Function (DF) - Deferred function waits are caused by a resource within the Control Program that must be serialized. Further examination of deferred function waits can be done by estimating the percentage of time each lock is busy. Taking the square root of the percentage of spin time per second for each lock will give an approximation of how long each lock is active. A separate lock report estimates the time active for the system lock, the freelist lock, the run list, the timer request lock, dispatcher lock, I/O lock and the real memory lock. The busiest of these is likely to be causing the deferred function wait.

Instruction Execution (EX) - If this wait is high, the instructions which are being simulated most often will offer clues on what is causing the wait. A separate report lists the number of occurrences for each privileged instruction execution per second for each interval throughout the day.

Terminal I/O Wait (TIO) - This is a special case of the privileged instructions wait in which terminal I/O is being waited on. This is determined by the TIO bit being on at the same time the user is in instruction execution wait. A high percentage of users in terminal I/O wait may be caused by many full-screen reads/writes instead of reading only changed data.

Run User (RU) - This is a user who hasn't yet been dispatched and so cannot technically be counted as CPU wait.

PSW Wait (PS) - This is a user which has entered a voluntary wait state and, at least for the first-cut, does not necessarily represent badness. For this reason the number of users in PSW wait was subtracted from the in-queue users for the denominator when calculating the percentage of in-queue users for each of the above wait states.

Unknown Wait (UK) - Finally if a user is in-queue but failed to fit in any of the above wait states, the state is called unknown. This should be the exception.

The above information is readily available in real-time with the 'IND POS' or the 'IND Q' command and has proven to be an effective measure if the system has just bogged down. These displays give the numbers in most of the above wait states and response is much faster than going into a 'real time' monitor.

5. Real Storage Indicators

Storage was the next resource looked at closely since VM is generally very slow when 'serious' problems occur within its storage subsystem. High activity on the real-memory lock mentioned before as a deferred function wait will give a clue to storage problems, but other variables need to be examined as well. The VM system which prompted this paper had a high number of page frame moves above and below the 16 Meg boundary. The cause was later determined to be from a high usage of IUCV (Inter-User Communication Vehicle).

Paging and swapping in VM must happen quickly if response times are not to suffer. Tetzlaff and Beretvas [2] have written a good paper summarizing paging in VM, so only a couple of sentences will be restated here. When more real page frames are needed, the first place VM looks is the non-interactive queue for users which have exceeded the SRM value SWPQTIME. Next the flush list is made available (these are users' pages which were not referenced during the last queue stay). The next place to get page frames is from the logically swapped users, first the non-interactive and then the interactive that are currently not in queue. The last resort is a corescan which takes pages from in-queue users. This presents a problem for VM and thrashing can occur if this stealing happens often. The following variables, page frame steals per second, pages examined for stealing, and full corescans per second, give an indication how much page stealing is hurting the system.

Since the storage service time depends on how fast pages can be brought in and out of memory, the speed and size of primary paging is important. A rule of thumb that 80% of the active users working sets would fit in the primary paging area was used to determine the best size for primary paging. However, the value of 'active users' is actually a function of time. The longer the time period, the larger the number of active users hence a larger total working set size. Active users can be reported on for any time frame equal to or larger than the monitor interval. This implementation used a five minute time frame to determine primary paging size. Active users are easily counted in REXX from the user records by using 'associative memory' or 'content addressable' features of compound REXX symbols (arrays). This implementation calculates several active user counts rangeing from a time length of each monitor interval to an hour.

6. CONCLUSION

What was learned from all of this? A purchased model of VM would not answer many performance questions for this particular system. This system was special in the sense that most of its workload consisted of 25 TPF operating systems and VM did not see enough detail of what was happening within each system for the model to yield accurate results. Using measured response times along with data from TPF system collectors, enough detail may be able to be supplied to the model to make it converge, but the manual effort is great.

The knowledge that contention analysis for an existing VM system could be preformed without a great effort provided the motivation for writing this paper. The analysis which was performed on a currently running VM system has been documentated. However, there is still the very big short-fall of not being able to project what changes will do to response time. Contention analysis will identify hardware shortages, but a model would be useful for projecting response times with these hardware changes. To make this model reasonable to use, more data will be required about the guest operating systems.

REFERENCES

[1] Koeppel, K.W. P., Measuring VM Systems - An Introduction To A Systems Perspective, CMS Transactions, No. 53, 1986, pp. 27-40.

[2] Tetzlaff, W. and Beretvas, T., Paging in VM/370 Operating Systems, CMS Transactions, No. 53, 1986, pp. 65-76.

[3] Virtual Machine Monitor Analysis Program, User's Guide and Reference, SC34-2166, IBM Corporation.

[4] VM/370 System Programmer's Guide, GC19-6224, IBM Corporation.

APPENDIX A

```
PARSE VALUE DIAG(14,'RNMNSFB','00C',15) WITH CC 2 . 3 RYP1 7 . 9 SFB
DO FOREVER
   PARSE VALUE DIAG(14,'RNMNSB , 00C') WITH CC 2 . 3 RYP1 7 . 9 BLOCK
   IF CC =1 THEN LEAVE
   BLKLEN=C2D(SUBSTR(BLOCK,17,2))-4
   INDEX=21
      DO WHILE INDEX < BLKLEN
      RECLEN=C2D(SUBSTR(BLOCK,INDEX,2))
      RECORD=SUBSTR(BLOCK,INDEX+4,RECLEN)
      CODE=C2D(SUBSTR(RECORD,1,3))
      SELECT
        WHEN CODE = 97 THEN DO
        .
        .
        .
        END
     .
     .
     .
     END
   INDEX=INDEX+RECLEN
   END
END
```

APPENDIX B

```
     VMRSTAT=SUBSTR(RECORD,57,1)
     VMDSTAT=SUBSTR(RECORD,58,1)
     VMPEND=SUBSTR(RECORD,68,1)
     VMSWSTAT=SUBSTR(RECORD,82,1)
     SELECT
       WHEN BITAND(VMDSTAT,'02'X) = '02'X THEN E=E+1  /* ELIGIBLE */
       WHEN BITAND(VMDSTAT,'04'X) = '04'X THEN SELECT /* IN A QUEUE */
            WHEN BITAND(VMPEND,'80'X) = '80'X THEN DF=DF+1
            WHEN BITAND(VMSWSTAT,'80'X) = '80'X THEN SW=SW+1
            WHEN BITAND(VMRSTAT,'40'X) = '40'X THEN PG=PG+1
            WHEN BITAND(VMRSTAT,'20'X) = '20'X THEN IO=IO+1
            WHEN BITAND(VMRSTAT,'08'X) = '08'X THEN DO
               IF BITAND(VMDSTAT,'10'X) = '10'X THEN TIO=TIO+1
               ELSE EX=EX+1
               END
           WHEN BITAND(VMRSTAT,'10'X) = '10'X THEN PS=PS+1
           WHEN BITAND(VMDSTAT,'08'X) = '08'X THEN DO
              IF BITAND(VMDSTAT,'80'X) = '80'X THEN CP=CP+1
              ELSE UK=UK+1
              END
           WHEN BITAND(VMDSTAT,'80'X) ¬= '80'X THEN RU=RU+1
           OTHERWISE NOP
           END
      OTHERWISE DO  /* NOT IN A QUEUE */
          IF BITAND(VMRSTAT,'80'X) = '80'X THEN CF=CF+1
          ELSE IF BITAND(VMRSTAT,'01'X) = '01'X THEN IDLE=IDLE+1
          ELSE DO
          NOP=NOP+1
          END
          END
       END
     END
  OTHERWISE NOP
END
```

END-USER RESPONSE TIME MEASUREMENT - NOT THE FINAL SOLUTION

Per Futtrup
Morten Panduro
Bent Vestermark
Scandinavian Airlines System

ABSTRACT

In recent years efforts to secure the end-user a proper response time have had a tendency to focus on tools for measuring this. Once the tool is in place you will find that the figures obtained only tells you whether or not they are acceptable. If they are not, you will find that you need further tools to pinpoint problem areas. Therefore tools for problem determination should be based on systems that track parts of the total system instead of end-user response time.
The tool for end-user response time measurement is still an excellent tool for management reporting, and to ensure that end-user response time is acceptable when none of the other tools reports any problems. In this session we will present the principles of tools in use at our installation, and our experience with these tools.

1. LIMITATION.

At our installation we work with three concepts to characterize the service delivered to the end-user:

- response time
- unavailability
- instability

Instability is the grey area between normal operation and unavailability, which includes things like retransmissions, load regulation and fallback mode.
These three concepts are rather well defined from a technical point of view, but are very difficult to communicate to the users.
The question raised by a user which above all made us aware of this problem was the following:

> 'where is the limit between long response times and unavailability'.

With hot standby for the communication computers a stop normally takes less than one minute, and with a non technical user this may look like long response times. This problem remains unsolved at our installation.

In this session we will concentrate on tools used to secure the end-user an acceptable response time. Although instability will influence response times, we will not cover that, and we will not cover response times in fallback mode either.

2. INTRODUCTION.

The high importance of today's online systems has caused a great interest in service delivered to the end-user, and thereby also in tools for end user response time measurement.

We have had such a tool in use for some time, and have found that there are several problems involved, if problem determination is based on such a tool. Some of them are:

- Several components are involved in the path from the end-user to the application: terminal line, communication computer, trunk line, etc. There are a lot of places to search for the reason.
- In our company three different departments are involved, and as Murphy says, 'If more than one person is responsible for a miscalculation, no one will be at fault'.
- If the reason for the problem is in the application or communication computer, you may have thousands of users on your list with response time problems.

We find it better to prevent than cure, and therefore we think that tools used for problem determination should be tools that track and report on individual components in the system, and not on complex entities, which the end-user response time is.

2.1 SPLIT OF SERVICE LEVEL.

This means that the service level for end-user response time must be split into service levels for parts of the system, not necessarily for each component. We have split our service level into three parts, as can be seen in figure 1.:

- Host.
- High Level Network which includes interface, front end, trunk line, and remote concentrator.
- Low Level Network which includes terminal line, controller and the terminal.

Communication with users is still done in terms of end-user response time. The split was done as part of an effort to improve service through a better motivated staff within the DP departments, and it is not without problems.

It is rather easy to define host service level for an application, but fulfillment of this may only fulfill end-user service level for a part of the total user group. some users may have a worse service because of the way they use the application.

Host service level is normally defined for peak periods, e.g. peak hour, and this period is in most cases well defined and well known. As you move out from the host, the peak periods will vary more and more, especially if you have a world wide network. This means that you do not have one peak hour any more, but a lot of them.

The split has the advantage that each individual DP department has its own goals to fulfill, and by that also the possibility to generate exception reports for their area. Also they can generate reports showing their performance against their obligations, and by that motivate staff to improve service where needed.

The system highly simplifies the problem determination, because reports are sent directly to the responsible department, and only reports needed are generated.

2.2. SPLIT RESPONSIBILITY.

In some installations the DP department is only responsible for parts of the total system between the end-user and the application, although they may be responsible for the end-user response time. In these cases it is highly important that they know what service their part of the system is giving.

Figure 2. shows an example where their responsibility ends at the connection between the terminal concentrator and the foreign system.

Figure 3. shows an example where a foreign system is put in between the data department's remote concentrator and terminl equipment. In this case measurement of end-user response time (in our own equipment) in combination with knowledge about our own service, gives the possibility to report on service received from the foreign system.

We have all abovementioned cases in our total system.

2.3. MANAGEMENT.

Comments to management reports based on measurements of end-user service are more easily given if each department has its own follow-up system with historic exception reports giving:

- the reason for the problem.
- what was done to remove the problem.
- what was done to ensure that this will not happen again.

In the following we will present the tools in use at our site. It should be mentioned that the network basis for this session is a network based on UNISYS hardware and software.

3. END-USER RESPONSE TIME MONITOR.

When we implemented new terminal equipment in early 1986, it became possible to measure the real response time at the end-user. The terminal equipment can save measurements against 4 different applications. For each of these applications we have 4 response time limits with corresponding counters, plus an overflow counter that counts transactions with response time higher than the highest limit. Besides that we have a counter for total resonse time. The response time can be measured in three ways, from pressing the send button until first character appears on the screen, or until ACK received on input message (not end-user response time), or until cursor is released.

3.1. PRINCIPLES.

We only had a few measurements of end-user response time before this equipment was installed. During installation we decided upon the following principles:

- The new equipment should not be installed at strategically important places until it was reasonable stable.
- All terminals in a work area should be exchanged with the new type to avoid confusing the users.
- To get as broad a picture of response times as possible, we exchanged equipment both in various geographical areas, and in areas where different applications were used.
- The points chosen in the initial phase are used in monthly reports to be able to compare figures from month to month.
- The applications to be measured for a terminal shall be:
  application 1 - most used application.
  application 2 - decided by the user.
  application 3 - decided by the user.
  application 4 - all other transactions.
- Response time limits were universally set to 3 sec, 4 sec, 7 sec, and 12 sec.
- Response time definition was universally set to 'release of cursor'.
- Collection of measurements to central site is done once every 24 hours. They can be collected more frequently if needed.

3.2. USAGE.

The tool is mainly used for the following:

- Reports on service to management and users.
- To determine if the end-user service level is fulfilled when service levels for all parts of the system are fulfilled.
- As mentioned above to control service delivered from a vendor.
- To help users complain about bad response times. Measurements can be displayed at the users terminal, so he can determine whether he gets a service according to service levels.
- To help technicians to see the effects of certain changes, e.g. if traffic is sent through a statistical multiplexor.

3.3 REPORTS.

We try to satisfy management's and end-user's need for information through our monthly reports. We have no fixed reporting structure for technicians, so their need is covered through adhoc reports. The monthly reports consist of two parts, a graphical part which reports on the initially chosen points, and a text part containing data about all terminals being measured, today about 850. The graphical part consists of two types of reports.

The first report, shown in figure 4, gives an overview of a month's service. The left part describes how large a percentage of the terminals had fulfilled service-levels. The right part shows how much the total average response time deviates from present goal. Application MP had a total average response time half a second better than goal, but servicelevels was only fulfilled for 50% of the terminals.

The second report, shown in figure 5, gives response times per application, and shows the development over the last 6 months. The left part gives response time distribution within 4 limits. The right part gives the average response time for the whole month, and for the worst day during the month. Besides these reports we have a menu driven interface against the database, which gives our front office and others access to detailed data and allows them to compose reports in various ways.

The service levels mentioned above were set internally in the data department during implementation of the monitor, due to lack of service levels agreed with the users. Agreements have later been reached for two major applications, and they say that 90% of the transactions shall be below 4 seconds.

4. HOST.

The definition of host response time is the time from message receipt on input queue until the response is placed on output queue. Service level is defined for two of our most important applications, and says that 90% of the transactions shall have a host response time below 1.3 seconds.

Daily monitoring of response times are done for applications and transaction codes, but not for individual users or groups of users.

Figure 6 gives an example of a report from the host response time monitoring system.

5. HIGH LEVEL NETWORK.

The service level for response time in the HLN is that 90% of the transactions shall be below 0.3 seconds. The response time in this part of the system is defined as the time from when the input message is totally received from the terminal line until the output message is placed on the queue to the terminal line, minus time spent in the host. The service level is in principle only defined for two major applications. The average message length influences the response time, but with the line speeds and message lengths currently used in our installation (minimum line speed 64 KB and average output message length 250-600 characters), this will be valid for almost all applications.

Follow-up includes communication computers, trunk lines and interfaces to hosts. We have further implemented a special response time monitor for the high level network.

5.1. COMMUNICATION COMPUTER FOLLOW-UP.

Continous check of communication computers is based on CPU and buffer utilization. Number of open/close sessions and disk accesses are also important for the performance. Data for this are collected continuously, but only used on an ad-hoc basis, because unfavourable development of these figures will be reflected in the CPU utilization.

5.1.1. CPU UTILIZATION. Monitoring of CPU utilization consists of checking two things:

- that the load on the CPU (traffic) develops according to plan.
- that the capacity of the communication computer (maximum traffic that can be handled per second) does not decrease, e.g. because of changes to software.

Two reports are produced and checked daily. The first report, shown in figure 7, is a plot containing the previous day´s observations with CPU IDLE % along one axis, and traffic in messages per second along the other. This plot shows traffic load during the day, and whether or not abnormal incidents have happened. On this particular plot a group of observations can be seen down to the right, representing a load much higher than expected. A detailed analysis showed that the traffic between the communication computer and the development system was 50 segments per second higher than normal, probably caused by a looping message. In this case the maximum load stayed below the maximum capacity (the reference line at 15% CPU IDLE) so no harm was done to the users.

The second report, shown in figure 8, gives the capacity of each communication computer calculated on that day´s observations. Further, the peak traffic rate is shown. We operate with two limits for the CPU. The first at 30% CPU IDLE is the planning limit, and is used when plans are made one to three years ahead. The second is at 15% CPU IDLE, and this limit represents the maximum usable capacity.

Another report, shown in figure 9, is produced monthly. This report shows both utilization of the computer and capacity in messages per second this month compared to the previous months, by means of regression lines. It can be seen in this example that capacity has been about 15% lower than 6 months before, but that it is moving back up again. The reason was mainly implementation of new discs, but tuning has been done so the capacity will come back up where it was before the implementation.

5.1.2 MESSAGE BUFFER UTILIZATION. Control of buffer utilization simply consists of ensuring that an adequate number of buffers are available at all times.

The report shown in figure 10 is created daily, and shows average and minimum number of free buffers in 10 minute periods (recording interval) throughout the day. The figures should be above the reference line at 3000 free buffers.

The smallest daily figures for average and minimum number of free buffers are collected and presented in the monthly report shown in figure 11. This report covers a period of 3 months.

5.2 TRUNK LINES AND INTERFACES.

Delay in these parts of the system depends on line utilization, line speed and message length when hardware and software (protocol) has been selected.

Reported service level for the high level network is partly based on experience from the response time monitor in the high level network mentioned below, partly on calculations of delays in the present configuration.

Maximum utilizations from these calculations are used in the follow-up system for the lines. Experiments have shown that we can go even higher than these utilizations without significant impact on service.

The maximum utilization has been supplemented with a warning limit placed 10% lower than the maximum, to have time for corrective action before service levels are exceeded. The follow-up system compares daily recorded peak trunk utilizations with the limits, and prints out an exception report if the alarm limit is exceeded. The reports can be printed on adhoc basis too.

Figure 12 gives an example of such a report for a trunk line. The alarm limit is set to 60%, and the maximum capacity to 70%. The plot shows the range of the three highest observations per day measured over 10 minute periods, both for in-going and out-going traffic. Exception reports are created based on the largest traffic in each direction, because the trunk line is a full duplex connection. The traffic has not passed the alarm limit in the report shown.

5.3 HIGH LEVEL NETWORK RESPONSE TIME MONITOR.

Early in 1987 we got the opportunity to measure response times in the high level network. This monitor can measure against up to 8 destinations, and it is based on synthetic messages instead of real traffic.

We measure against our 5 main hosts and our two most important applications. Each node in the network sends out messages to the above mentioned 7 destinations, and measures the time from when the message is sent until it is received again. The message turns around in the communication part of the host when hosts are measured, whereas measurement of applications are done by executing the same transaction again and again.
It should be noted that this software has a low priority in the communication node, which means that under abnormal conditions it will not be executed.

The resonse time limits are, after a couple of adjustments, set to 0.1, 0.3, 0.7, and 2 seconds. In reality we have one more limit, because the system times out after 20 seconds, so messages with a higher response time are counted as ´no response´. This means that the overflow counter contains messages with response times between 2 and 20 seconds.

Before implementation of this tool we had calculated the average peak response time to be 0.285 seconds. The first month we measured an average response time of 0.257 seconds. This was measured off peak, and around 30 milliseconds has to be added to adjust for message length.

The reports produced from this system correspond mainly to those mentioned under end-user response time, with a graphic part and a text part.

The graphic part has one page per main host or application (figure 13), which shows the response times for the previous month. The left part shows the response time distribution from remote concentrators against the PRODC host (plus ALL which is the distribution of all response times measured). The right part shows average response times for the whole month and for the worst day.

The text part also gives response times from front ends, plus the number of messages which is the basis for the calculations.

6. LOW LEVEL NETWORK.

The low level network is the only part of the system where we cannot directly measure the response times. We can estimate these by subtracting host and high level network response times from response times measured by the end-user response time monitor.

The service level agreement, which presently exists only for the two major applications, says that 90% of the transactions shall have response time in this part of the system less than 2.4 seconds. This means that half of the total response time may be caused by the low level network.

The definition of response time here is time from depression of the send button until response is presented on the screen and the cursor is released, minus the time from when input is totally received by the communication processor until a response is placed on output queue in the processor again.

Follow-up in this part is based on the fact that line utilization can be translated to response time, when we disregard instability like retransmissions.

The follow-up system consists of two parts:
- continuous control of line utilization
- modelling to estimate maximum acceptable line utilization when service levels shall be fulfilled.

6.1. LINE UTILIZATION.

The continuous control of line utilization is similar to the system mentioned under trunk lines and interfaces. The maximum and alarm limits vary more because of differences in configurations. The system has only been used for a few months, and we may have to change the criteria for creation of exception reports. Reports are created when the three highest observations for a day (based on 10 min. periods) pass the alarm limit; however this criterion may change based on experience.

Figure 14 shows an example of an exception report. This report was produced in the initial phase of the system, and has only one limit set very roughly. This was a line with much too high a line utilization.

The follow-up is based on total traffic on the line (characters per second), not subdivided in any way e.g. per application, and assumes that no major change of message length will occur (application programmers are expected to warn us about that).

6.2. MODELLING.

Maximum and alarm limits are estimated by means of a modelling tool when existing lines have to be reconfigured or when new lines are planned.
More detailed information is needed for this job than that used in connection with control of line utilization.

The report shown in figure 15 is created when lines are reconfigured. One week's data are the basis for the report and it gives the three highest observations during that week, based on total characters for the controller. The report can also be created at times where the line peaks.

The traffic to and from the controller is only subdivided into equipment types (equ), but we may have to subdivide it into applications too in future, because service levels are agreed per application.

The information is put into the model, and it is decided what action to take, normally either increase of line speed or decrease of traffic by moving one or more controllers to another line. After that, the maximum line utilization is determined.

For new lines traffic may not be available, and therefore the figures are estimated based on knowledge about what the terminals will be used for.

6.3 CONTROLLERS.

One assumption in our modelling work is that the controllers cause no delays, but increasing line speeds, new communication computers, and increased use of complex functions like cursor positioning and protected fields have changed this.

We saw the problem first time 2 to 3 years ago, when an increase of line speed caused longer response times. The reason is that the microprocessor in the controller is too slow, and we are for the moment running some load-tests, to find the configuration limits we have to observe to ensure that controllers cause no delays.

7. COMPARISON OF MEASUREMENTS.

To compare figures from the various tools we have to keep in mind what is measured and over which periods. All tools used for problem determination are based on peak periods, whereas response time monitors are based on 24 hour periods. The following summarises that:

Tool	what	period
End-user response time measurement	transactions per terminal and application	24 hours
Host response time measurement	transactions per application	peak hour
High level network response time measurement	sample messages (length 10 chars)	24 hours
Communication computers	cpu and buffer util	10 min. periods all 24 hours
Trunk lines and interfaces	all user traffic	10 min. periods all 24 hours
Low level line utilisation	all user traffic	10 min. periods all 24 hours
modelling	several possibilities	peak hour

The end-user service level, 90% below 4 seconds, are defined for 24 hour periods, so this fits with the response time monitors.

The goal for each DP department involved is defined as a percentile, so we must be lucky if the end-user goal is fulfilled when all three parts are fulfilled. To specify the goals as percentiles in stead of averages has made the whole job more difficult, but we had no influence on that decision.
The modelling tool used for the low-level network can only report average response time, so we have no knowledge about percentiles here. To overcome this problem we have looked at several places with end-user RTM, to estimate an average low level network response time which can replace the percentile.

In the following we have specified some measurements. We have specified average figures, the only figures available for the low level network, to be able to compare with the end-user RTM, but the following is valid:

Host: 90% or more is below 1.3 seconds, when the average is below 1.0 seconds.

High level network: 90% or more is below 0.3 seconds when the average is below 0.3 seconds.

The partial figures represent different things, so the sum of these will not add up to the average shown under end-user RTM. We expect the sum to be higher because the host figure and the figure for the low level network is based on peak periods.

line S1J against the CSS application

partial figures.

date	host	high level net RTM.	low level net. modelling	
	avg.	avg.	avg.	%util.
27apr87	0.59	0.24	1.74	18.2
28apr87	2.05	0.25	1.42	13.5
29apr87	0.62	0.25	1.44	10.0
04may87	1.04	0.18	1.33	13.4

total figures.

date	end-user RTM.		sum of partial figures.	difference
	%<4	avg.	sum	%
27apr87	95.2	1.97	2.57	30
28apr87	94.6	2.07	3.72	80
29apr87	96.6	1.93	2.31	20
04may87	95.3	2.03	2.55	26

In this example the sum of the partial figures is 20%-30% higher than the end-user RTM, except 28apr87. This day the host had a problem causing the average host response time to be 2.05 seconds, but the end-user RTM shows that users on this line was not affected by the problem.

line D47 against the CSS application

partial figures.

date	host	high level net RTM.	low level net. modelling	
	avg.	avg.	avg	%util.
04may87	1.04	0.19	5.28	35.2
05may87	0.64	0.16	5.01	33.7
11may87	0.63	0.17	4.39	29.7
12may87	0.54	0.23	4.11	27.0
13may87	0.65	0.20	4.20	27.8

total figures.

date	end-user RTM.		sum of partial figures.	difference
	%<4	avg.	sum	%
04may87	70.3	3.75	6.51	74
05may87	73.4	3.47	5.81	67
11may87	72.9	3.43	5.19	51
12may87	71.1	3.62	4.88	35
13may87	72.4	3.48	5.05	45

There is a rather big difference for line D47, but that is because line utilisation was so high it caused a great difference between peak and off peak figures.

8. Conclusion.

We find that the end-user response time monitor is not mandatory to be able to obtain and maintain a certain service level for the end-user. We felt very uncertain about this until we got the tool, but we are now convinced that we can base our problem determination on the partial tools.

These tools are less accurate than the response time monitor because of certain approximations, but we find the results acceptable. One problem though is that users do not trust modelling in discussions about service, but we hope that this problem will disappear when we consistently deliver decent service, as there hopefully will be no reason for discussions.

Reports on measured end-user response time for all terminals cannot be produced, so this system do not satisfy management's need for that.

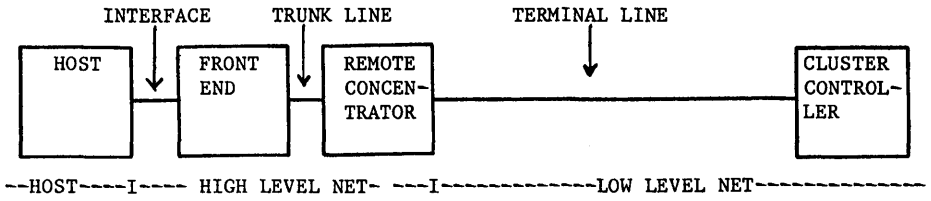


Figure 1.

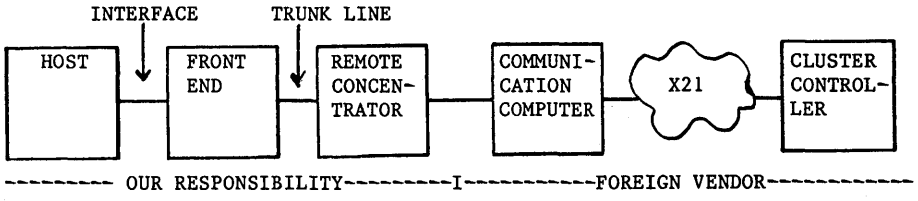


Figure 2.

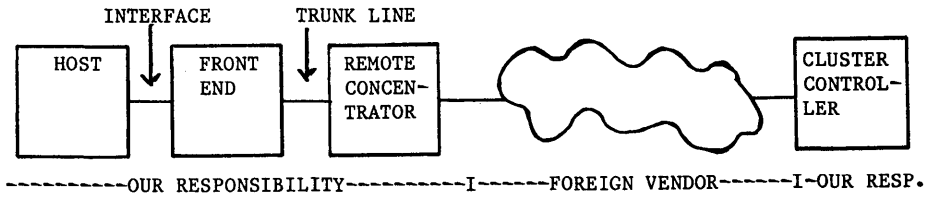


Figure 3.

COMMUNICATION MANAGEMENT SYSTEMS
PERIOD = APR87

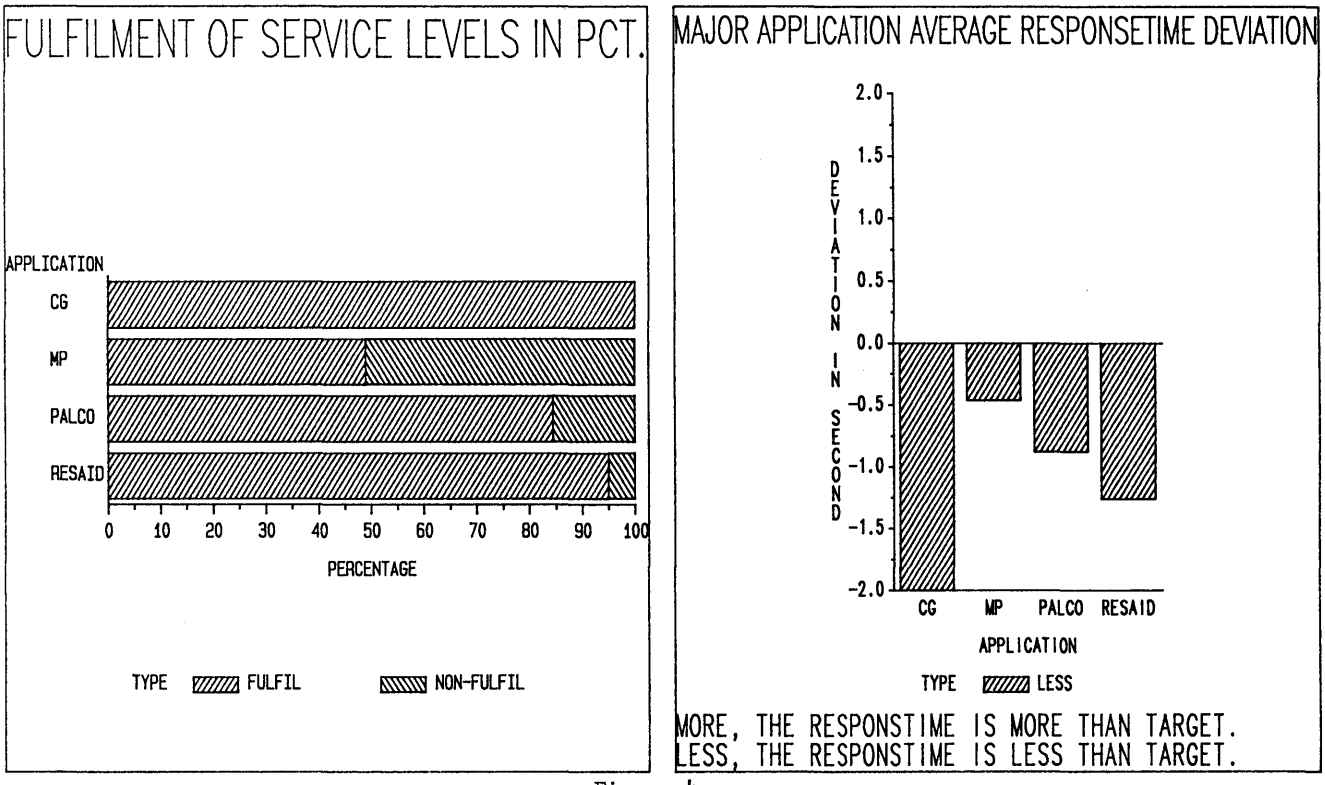


Figure 4.

CPHXN-N/1MAY87

PERIOD = NOV86 - APR87

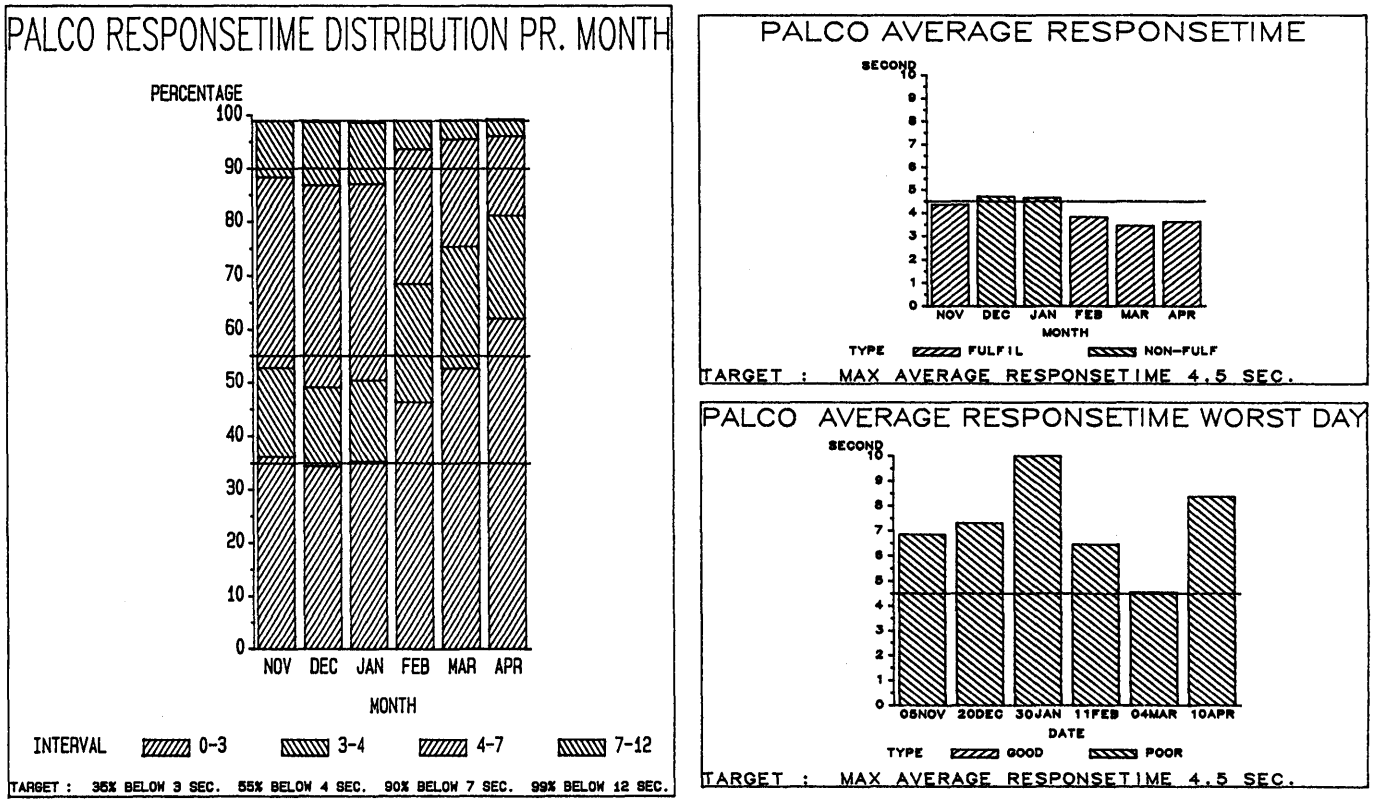


Figure 5.

CPHXN-N/1MAY87

CSS SYSTEM EXSTENCE-TIME IN MAINFRAME (10-11 L.T.)

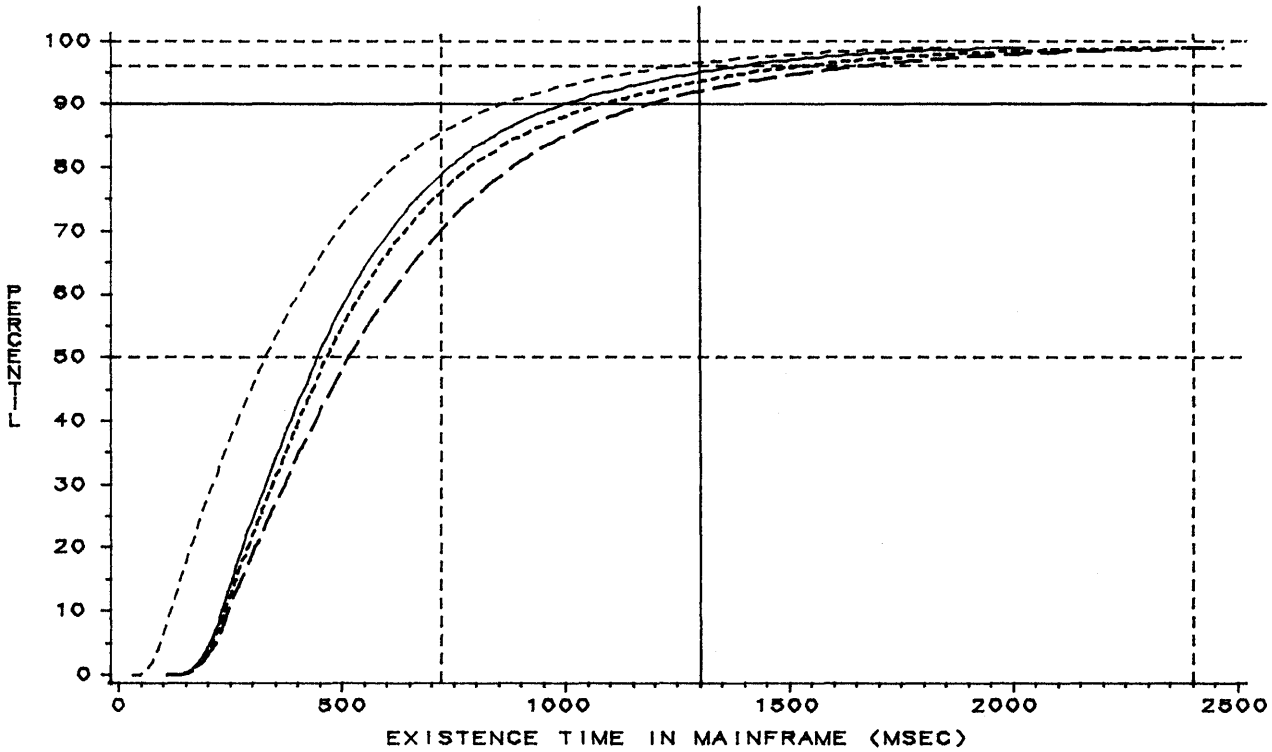


Figure 6.

DATE
24AUG87 61.5 TRX
25AUG87 58.3 TRX
26AUG87 56.5 TRX
27AUG87 56.0 TRX

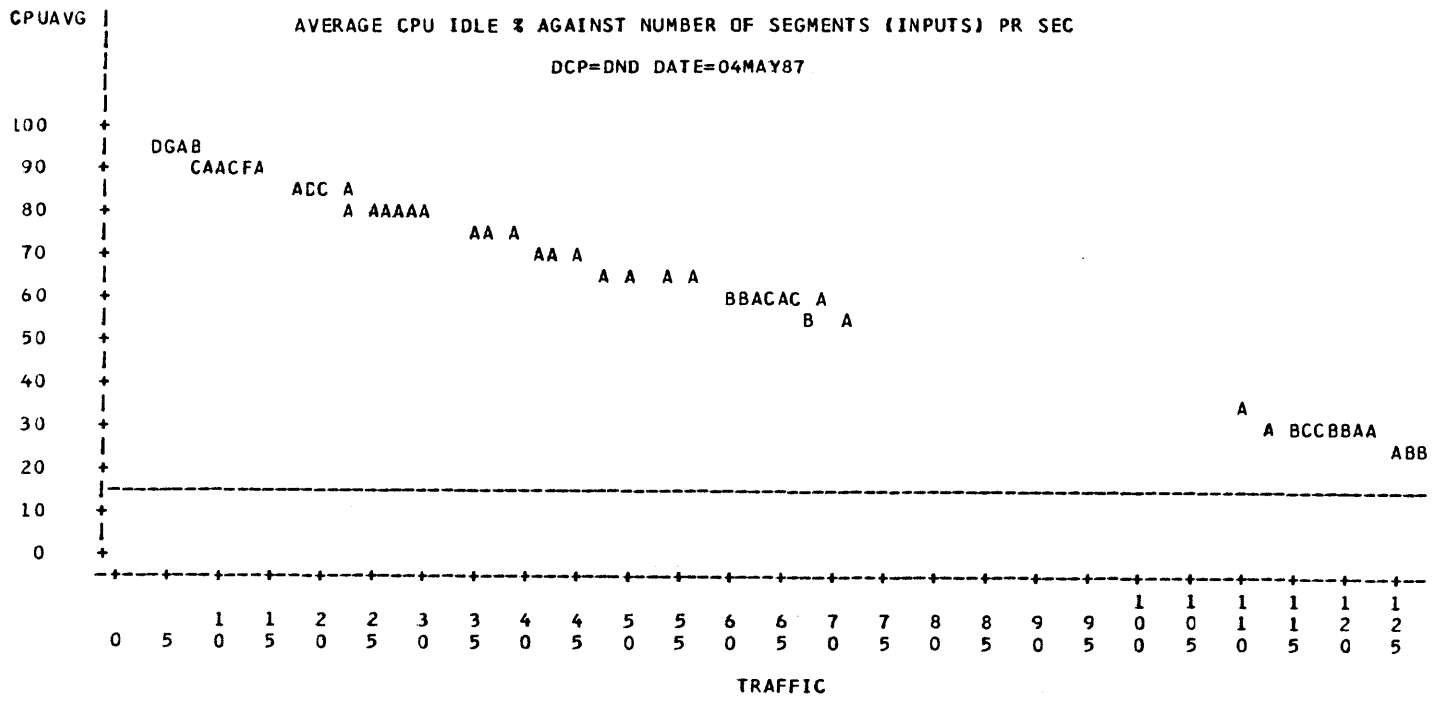


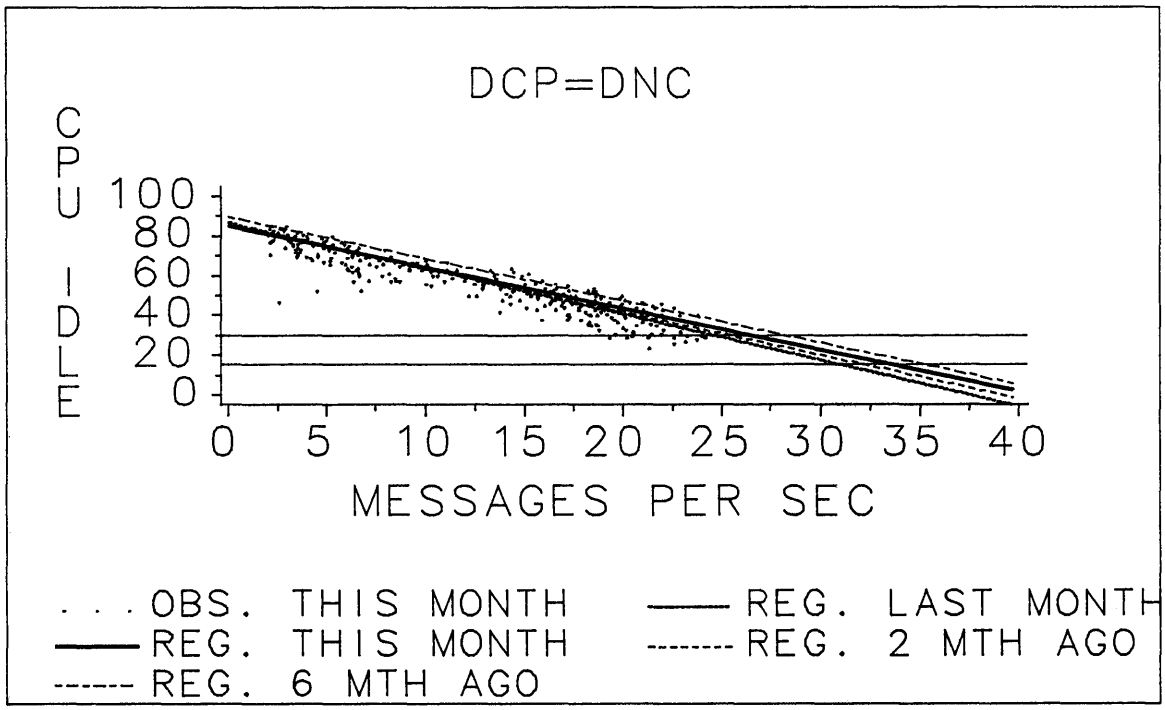
Figure 7.

ESTIMATED DCP CAPACITY

DCP	DATE	TYPE	CAP30	CAP15	PEAKRATE
DNA	11MAY87	FE	90.089	111.957	83.3817
DNB	11MAY87	FE	37.379	45.389	24.3383
DNC	11MAY87	RC	27.619	34.810	24.3933

Figure 8.

AVERAGE CPU IDLE %
AGAINST NUMBER OF MESSAGES PR. SEC.
MONTH=APR87



REF. LINES 15 = MAXIMUM CAPACITY 30 = PLANNING CAPACITY

Figure 9.

CPHXN-N/12MAY87

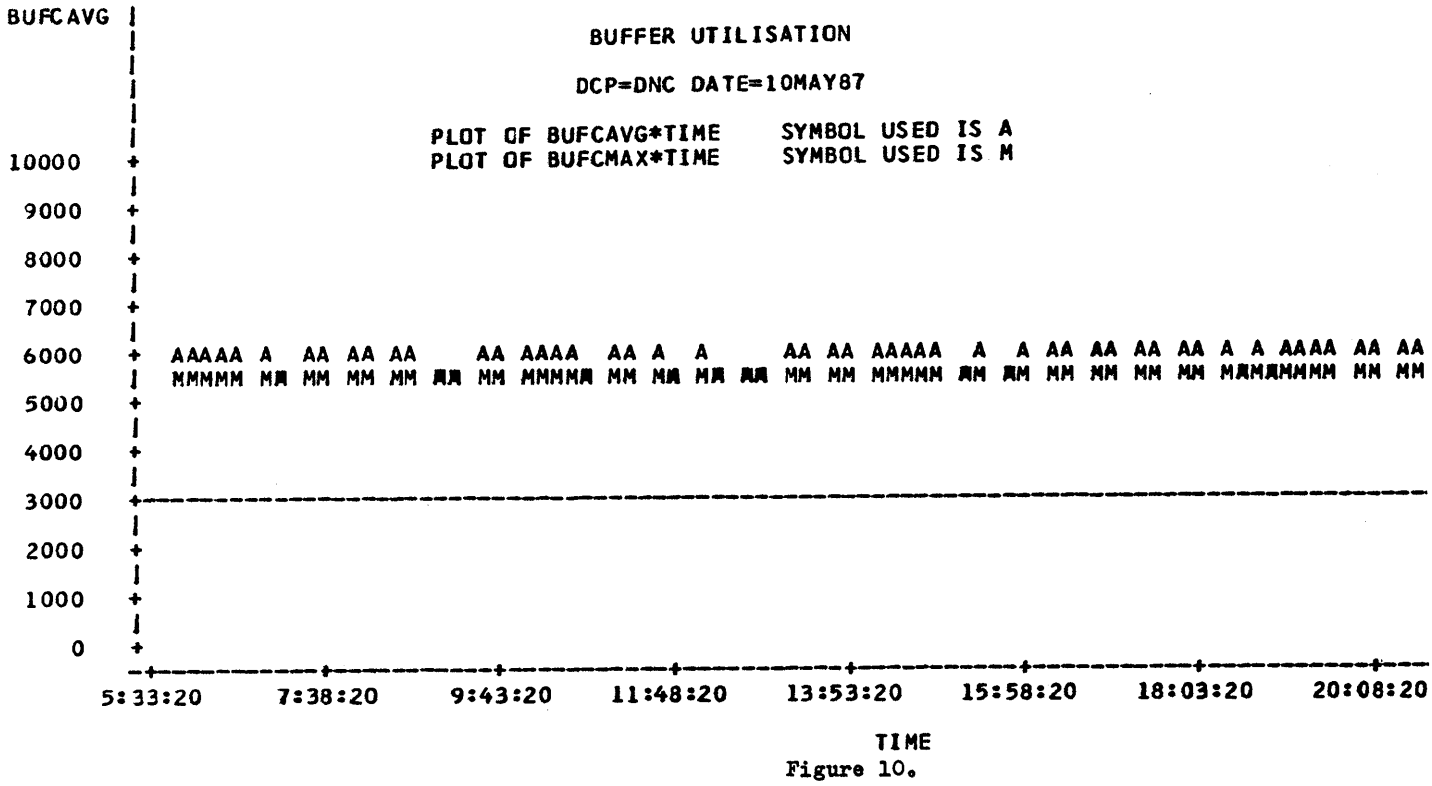


Figure 10.

COMMUNICATION MANAGEMENT SYSTEMS

BUFFER UTILISATION

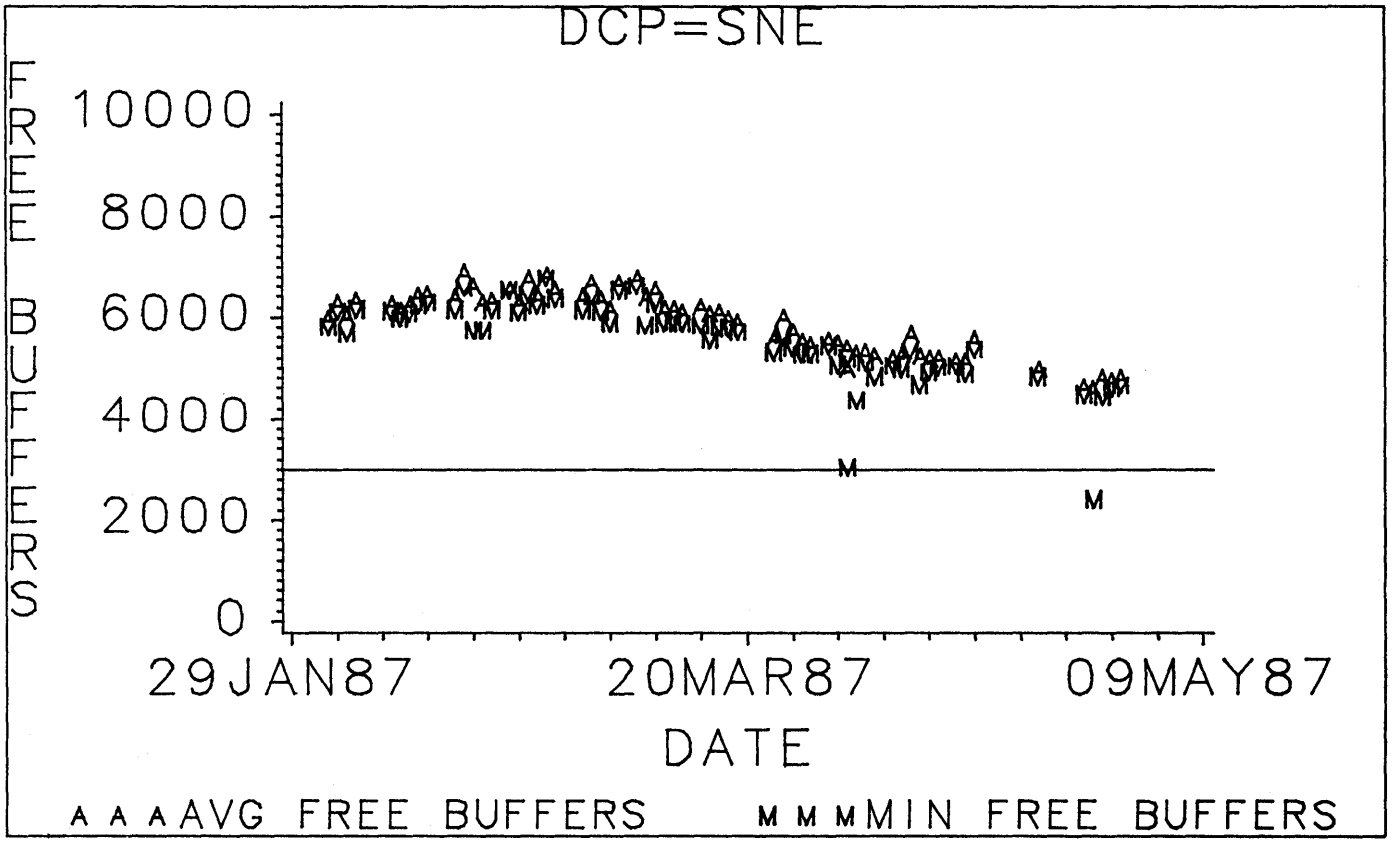
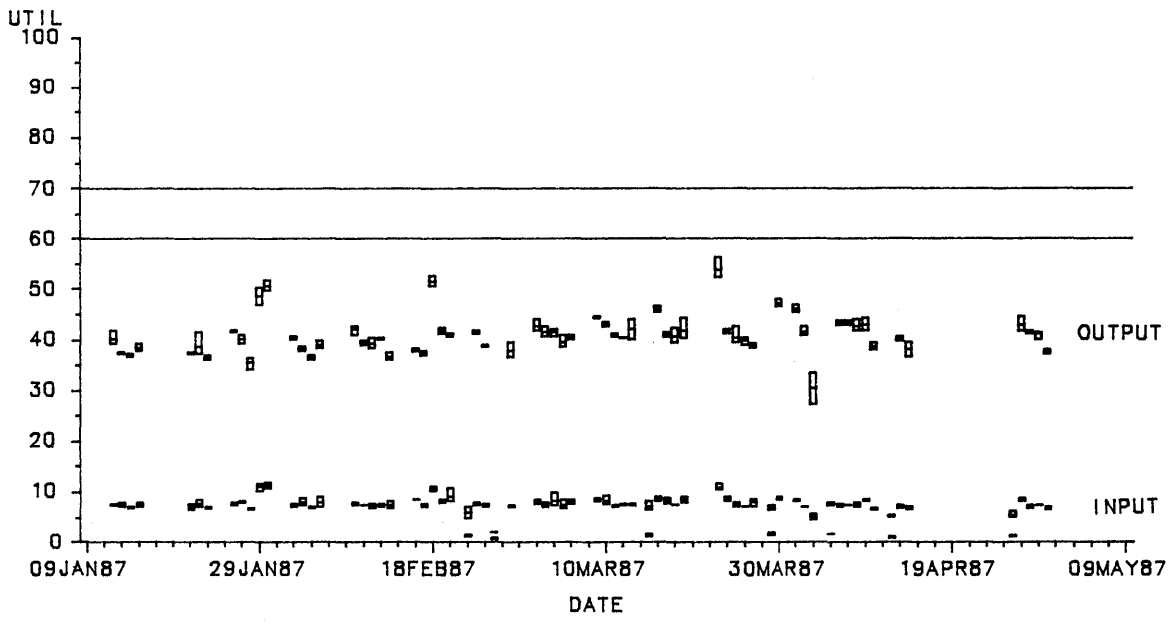


Figure 11.

CPHXN-N/11MAY87



SPEED IS 64KB
RANGE OF 3 HIGHEST OBSERVATIONS PR. DAY

Figure 12.

MAX CAPACITY 70%

PERIOD = APR87

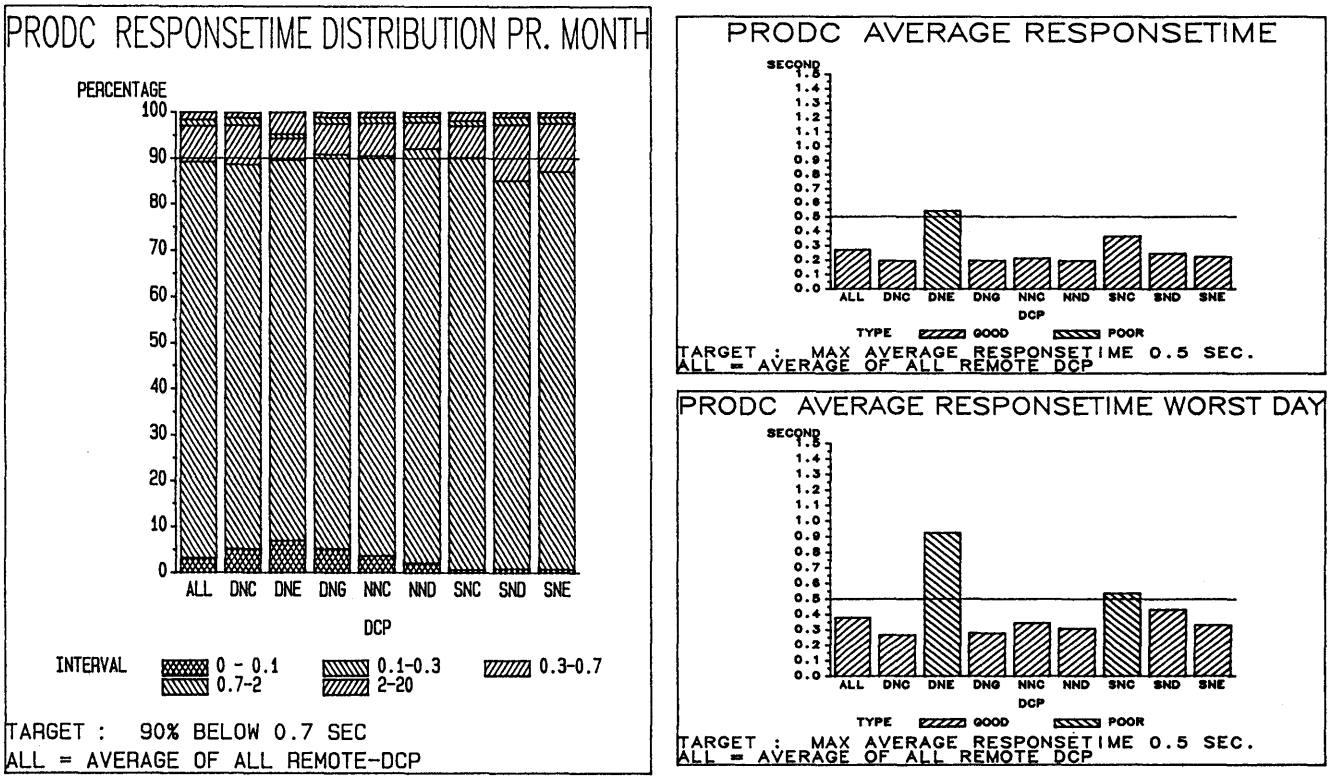


Figure 13.

CPHXN-N/1MAY87

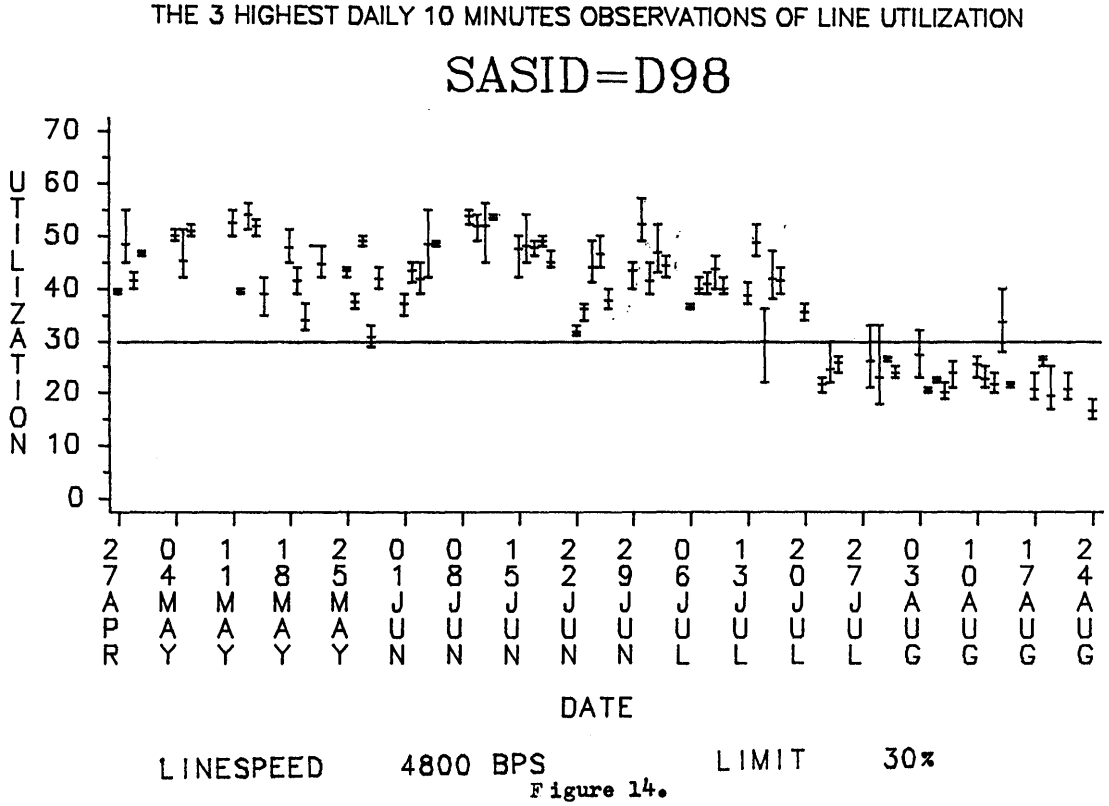


Figure 14.

PEAK HOURLY TRAFFIC FOR SELECTED CONTROLLERS

CTRL=D0861 DATE=31OCT86 HOUR=14

EQU	NBRTERM	NBRACT	MSGSNT	CHRSNT	AVGSNT	MSGRCD	CHRRCD	AVGRCD	TRANTERM
BPP	2	1	0	0	.	5	712	142	5
PRT	1	1	0	0	.	3	1632	544	3
TIC	1	1	0	0	.	14	5892	421	14
VDU	4	3	200	4740	24	185	53660	290	67
HOUR			200	4740		207	61896		

CTRL=D0861 DATE=02NOV86 HOUR=16

EQU	NBRTERM	NBRACT	MSGSNT	CHRSNT	AVGSNT	MSGRCD	CHRRCD	AVGRCD	TRANTERM
BPP	2	1	0	0	.	5	732	146	5
TIC	1	1	0	0	.	11	4476	407	11
VDU	4	3	152	4430	29	135	44660	331	51
HOUR			152	4430		151	49868		

EQU - Equipment type.
NBRTERM - Total number of terminals connected.
NBRACT - Number of active terminals during the period.
MSGSNT - Total number of messages sent.
CHRSNT - total number of characters sent.
AVGSNT - Average length of messages sent.
MSGRCD - Total number of messages received.
CHRRCD - Total number of characters received.
AVGRCD - Average length of messages received.
TRANTERM - Average number of transactions per active terminal.

Figure 15.

DETERMINING END USER RESPONSE TIME USING NLDM.

Michael W. Yount
Rohm and Haas Company, Philadelphia, PA 19105

ABSTRACT

Network Logical Data Manager (NLDM), now a part of Netview, is a software tool that collects end-user response time statistics from a Response Time Monitor (RTM). The RTM is a hardware feature that can be placed on 3274 control units and comes standard on 3174's. Unlike some other network software tools, NLDM reported response times are measured, not calculated response times.

The data collected by NLDM can be made available in several different ways. Of specific interest in this paper will be the ability of NLDM to write an SMF (type 39) record. The MICS SNA Network Component, SAS* software or some other tool may be used to extract the data from the SMF record. The data can be used to identify problem areas in the network, to report response time information to management or users and to set end-user response time objectives.

INTRODUCTION

Network Logical Data Manager (NLDM), now part of Netview, is an IBM program product. It requires Network Communication Control Facility (NCCF) and Virtual Storage Access Method (VSAM) as prerequisites. NLDM runs as an application program under NCCF and can collect and display three types of session data for System Network Architecture (SNA) environments:

- Session Awareness data
- Session Trace data
- Session Response time data.

This paper will discuss the area of response time data collection.

HOW NLDM WORKS

NLDM collects response time statistics from a Response Time Monitor (RTM). The RTM is a hardware feature that can be placed on 3274 control units and comes standard on 3174 control units. The control unit collects the response times into ranges of time that are specified when you define the performance classes in your NCCF parameter library (see List I).

The VTAM naming conventions your shop uses for applications, control units, and terminals plays an important role in the identification process. For example, our shop uses a naming convention for control units and terminals where characters 2-4 of the name identify the plant or office location, making the mapping of terminals to location a simple task.

Within the definition of performance classes you can set how the response time is to be measured. This is the time from when a transaction is activated (e.g. the ENTER key is depressed) until a response is received. This may be defined as:

- FIRST - When the first character of the reply from the host is received at the user's terminal. This is the default definition if no other is specified.
- CDEB - When an SNA 'change direction' or 'end bracket' is received at the user's terminal. (Because we use a significant amount of Session Manager, we found this to work best in our environment).
- KEYBD - When the user's terminal keyboard is unlocked.

NLDM collects the response time data from the control units by the issuing of a "COLLECT' command or at session termination. By taking advantage of NCCF services it is possible to issue the COLLECT command automatically using an NCCF timer-initiated CLIST. We found it desirable to collect the response time on 15 minute increments to correspond to RMF intervals.

NLDM can be configured to report response time in three different ways:

1) It will display response time trends and summaries during the day by collection intervals for a selected terminal. This can be useful in determining response degradation.
2) For terminals attached to a control unit with the RTM feature it can be configured to display the response time for individual transactions on the user's terminal.
3) The response time data is written out to an SMF record where it can be extracted and used for analysis, trending or for establishing and tracking end-user response time objectives. This can be accomplished through the use of an analyst created SAS program, by acquiring the SNA Network Component of the MICS data base or using code from Merrill's MXG package.

EXTRACTING THE DATA FROM SMF

NLDM creates type 39 (HEX '27') SMF records. The records created by NLDM may be any one of the following types:

Records	Identification Number
. Session start	3
. RTM collection	1
. Accounting and availability	4
. Bind Failure	6
. Combined	5
. Session end	2

Only the RTM collection, Combined and Session end records contain response time information. Some of the types of information that can be extracted are number of transactions measured, five response time collection buckets (user defined - see List I), the VTAM name for the application, the terminal ID, the performance class name (user defined - see List I), etc. Once the data has been extracted from the SMF records, it can be manipulated as required. For example, Table 1 shows a days worth of data by performance standards.

The first letters of the performance class (NSVPCLSS) name the application. The second part of the performance class is the location (e.g., CH - Cherry Hill, New Jersey). The columns that follow give the

*SAS is a registered trademark of SAS Institute Inc. Cary, NC, USA.

total transactions for that performance class followed by the number of transactions whose response fell within the counter boundries. This is followed by the percent this represents within each counter boundry. The last column shows the average response for the performance class. For example, there were 9066 CICS transactions from the Cherry Hill office. Of these 6474 or 71.4% completed in under 1 second, 91.0% completed in under 2 seconds, 95.2% in under 3 seconds and 98.1% in under 5 seconds. The average response time was 1.19 seconds.

VALIDATING THE DATA

For purposes of validating the data, displaying the response time on the user terminal affords the best opportunity for validation. The NLDM response times displayed at the terminal for individual transactions appear to be very close to the perception of the terminal user. For example, a logon is reported by NLDM to take about 15 seconds. Using the second-hand of a watch to verify this, the time is identical. Other transactions are too fast to verify with a watch but appear to be very close to the terminal user's perception of the elapsed time.

Trying to validate the data that is written out to the SMF records turned out to be a difficult task since the data must be compared on an "apples-to-apples" basis at the terminal level. Attempting to keep response time statistics at the terminal level requires a considerable amount of disk space. Also, the data collectors for most applications are not designed for summarizing the data to this level. However, with TSO/MON from Morino Associates, it is possible to collect host response times and sampled network delays for TSO to the terminal level.

Table 2 compares the TSO/MON response time statistics during peak hours (9:00-11:00 and 13:00-16:00) to NLDM's reported statistics during the same time period.

It is interesting to note the similarity, in most cases, between the TSO/MON sampled end-user response time and the NLDM measured end-user response time. This could be considered more a validation of the TSO/MON sampled network delay than of NLDM's measured response time.

Initially, it was expected that the host reported average response time and the NLDM reported average response time for locally attached terminals would be nearly identical. However, as can be observed from Table 2, this is not the case. Investigation has shown that this difference is primarily due to queueing delays within the 3274 control unit. According to studies performed by IBM (4), the control unit delay accounts for between .1 and .3 seconds for a full screen of data. These queueing delays appear to have been reduced on the 3174 control units, however, at the time of this writing, this had not been confirmed.

Remote users can expect further communications delays caused by line speeds, queueing in the FEP, number of drops on the line and other factors.

It should be noted that the largest variances that can occur are in the host reponse times. Depending on the type of transaction, host response times can vary from .2 seconds to 15 seconds or more. Network delays including control unit delays remain relatively constant across all transactions.

USING THE DATA

One of the uses for the data from NLDM is to set and report on end-user response time objectives. List II shows the table of response times compared to objectives (expected response) that we issue on a monthly basis.

This table, along with other important charts and tables are sent out in the form of a monthly report to several key users and managers.

The table shows the past three months' percent less than objective time and average response time for each location by application. Three months of data affords the user a clearer picture of the current month's data. A weekly version of this table is also produced and distributed to key MIS personnel so that any problems that arise can be dealt with as soon as possible.

Issuing a report on remote response times may not answer users questions completely. For example, one of the key top managers who receives this report was curious about the types of responses being received at Los Angeles. To monitor this location, a daily chart was produced (see example - List III) showing a breakdown of the transactions. This raised further questions as to the types of transactions that were greater than 5 seconds. By using the total response time as reported by NLDM and subtracting the average host response times we could determine the average network delay. Knowing this, we used the CICS component of MICS to identify those transactions whose host response time plus network delay were greater than 5 seconds.

Knowing the types of transactions that would cause long responses, the Los Angeles office personnel were instructed to inform the customer that a delay would occur. In some cases the customer was told that he would receive a call back when processing was complete.

In another study, similar charts were generated to prove that multi-dropping 56 Kbps lines would have a negligible impact on response times. This provided a cheaper alternative than adding dedicated lines and postponed the acquisition of an additional 3725.

COSTS

One of the questions that may have to be answered before using NLDM with RTM's is "Do I want to collect response time for all terminals in the network?". Depending on the size of your Network this may be impractical. We chose to select representative control units to be upgraded with RTM's. Another concern is that certain models of the control units (3274's) do not have options for adding an RTM. The cost to add an RTM is as little as $600, but may be considerably more if the control unit requires upgrading to a different model.

CONCLUSION

Using NLDM a measured end-user response can be obtained. Using an extraction tool, this data can be extracted from the SMF record created by this product and can be used to identify problem areas or areas in the network requiring upgrades, confirm or refute end-user response time complaints and to establish and report on end-user response time objectives.

REFERENCES

[1] IBM Corporation, Network Logical Data Manager Operation Document SC30-3165-2.

[2] IBM Corporation, Network Logical Data Manager Installation Document SC30-3287-1.

[3] Cook, J., "Communications Network Management - End User Issues", CMG Transactions, Spring 1986.

[4] Vondrasek, R., 3274 Performance Guideline Including 3178 3278 3279 and 3290 Terminals and the 3270 Personal Computer Attachment, IBM Technical Bulletin (ZZ20-4167-0), March 1983.

LIST I

The following is a sample member which would be defined within your NCCF parameter library. The definition statements in this member describe measurement parameters used by the RTM. There are two types of statements described in this member, the performance class (PCLASS) and a statement to map the session to the performance class (MAPSESS). Mapping of sessions to a performance class can range from all or a group of terminals as defined by a portion of their VTAM node name to a specific terminal (See PCLASS of TSOLCLT and MAPSESS of MAP2AT below).

```
CICCH    PCLASS      OBJPCT=80,+           --Objective Percent in this case 80%
                     OBJTIME=2,+           --Objective Time in this case 2 seconds
                     BOUNDS=(1,2,3,5,),+   --Collection buckets, 1 sec., 2 sec.,etc.
                     RTDEF=KEYBD,+         --How response is to be measured.
                     DSPLYLOC=YES          --Display response locally at terminal.
CICLA    PCLASS      OBJPCT=80,+
                     OBJTIME=5,+
                     BOUNDS=(4,5,8,10),+
                     RTDEF=KEYBD,+
                     DSPLYLOC=YES
CICLCL   PCLASS      OBJPCT=80,+
                     OBJTIME=1,+
                     BOUNDS=(1,2,3,5),+
                     RTDEF=KEYBD,+
                     DSPLYLOC=YES
TSOLCL   PCLASS      OBJPCT=80,+
                     OBJTIME=1,+
                     BOUNDS=(1,2,3,5),+
                     RTDEF=CDEB,+
                     DSPLYLOC=YES
TSOLCLT  PCLASS      OBJPCT=80,+
                     OBJTIME=1,+
                     BOUNDS=(1,2,3,5),+
                     RTDEF=CDEB,+
                     DSPLYLOC=YES
TSOSH    PCLASS      OBJPCT=80,+
                     OBJTIME=2,+
                     BOUNDS=(1,1.1,1.2,2),+
                     RTDEF=CDEB,+
                     DSPLYLOC=YES
*
MAP1A    MAPSESS     PCLASS=CICCH,
                     PLU=AA1CICS*,
                     SLU=A001T*      -- Sessions mapped to
MAP1D    MAPSESS     PCLASS=CICLA,    a control unit
                     PLU=AA1CICS*,
                     SLU=B053T*
MAP1E    MAPSESS     PCLASS=CICLCL,PLU=AA1CICS*,SLU=B001T*
MAP2AT   MAPSESS     PCLASS=TSOLCLT,PLU=TSO*,SLU=A001TF27 - To a specific terminal
MAP2A    MAPSESS     PCLASS=TSOLCL,PLU=TSO*,SLU=A001T*
MAP2B    MAPSESS     PCLASS=TSOSH,PLU=TSO*,SLU=B009T*
MAP2C    MAPSESS     PCLASS=TSOSH,PLU=TSO*,SLU=C009T* -- To all local control units
MAP5     MAPSESS     PCLASS=DFLTLCL,PLU=*,SLU=A*
MAP6     MAPSESS     PCLASS=DFLTRMT,PLU=*,SLU=* -- To all other control units
```

LIST II

RESPONSE TIME STATISTICS

THE FOLLOWING TABLE SHOWS THE REPORTED RESPONSE TIMES DURING PEAK HOURS (9:00-11:00) AND 13:00-16:00) * BY APPLICATION.

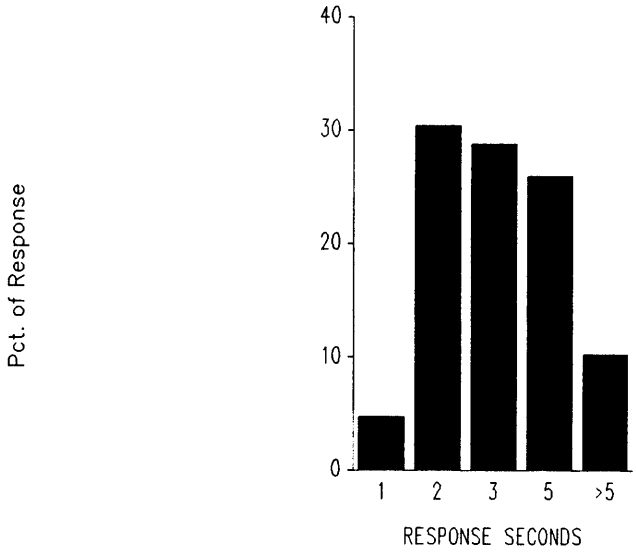
MAINFRAMES	PERCENT			OBJECTIVE	AVERAGE RESPONSE SECONDS		
	FEBRUARY	MARCH	APRIL		FEBRUARY	MARCH	APRIL
3084 TSO	79 %	78 %	78%	80 % < 1 SEC.	1.46	1.59	1.54
CICS	88 %	86 %	85%	75 % < 1 SEC.	0.60	0.74	0.74
4381 VM					1.10	0.90	0.75

LOCATION/APPLICATION	PERCENT			EXPECTED	AVERAGE RESPONSE SECONDS		
	FEBRUARY	MARCH	APRIL	RESPONSE	FEBRUARY	MARCH	APRIL
CHERRY HILL							
CICS	%	74 %	73 %	70 % < 1 SEC.		1.13	1.16
TSO	%	65 %	70 %	70 % < 1 SEC.		1.98	1.82
DALLAS							
CICS	78 %	75 %	76 %	ND < 3 SEC.	2.32	2.50	2.44
ENGINEERING							
TSO	91 %	89 %	89 %	ND < 2 SEC.	1.28	1.40	1.41
VM	79 %	87 %	94 %	ND < 2 SEC.	1.88	1.55	0.92
HOME OFFICE							
CICS	83 %	80 %	81 %	75 % < 1 SEC.	0.88	1.03	1.00
TSO	76 %	77 %	77 %	75 % < 1 SEC.	1.35	1.36	1.28
VM	82 %	86 %	93 %	80 % < 2 SEC.	1.42	1.46	1.00
LOS ANGELES							
CICS	78 %	75 %	71 %	65 % < 3 SEC.	2.52	2.73	2.87
SPRING HOUSE							
TSO	84 %	84 %	84 %	80 % < 2 SEC.	1.83	2.03	2.12
VM	79 %	83 %	86 %	75 % < 2 SEC.	1.72	1.61	1.41
VALBONNE							
TSO	80 %	71 %	74 %	60 % < 5 SEC.	4.56	5.97	5.75
VM	89 %	78 %	84 %	75 % < 5 SEC.	2.92	3.80	3.31

* VALBONNE IS BASED ON THEIR PEAK HOURS
ND - EXPECTED RESPONSE HAS NOT BEEN DETERMINED, MORE DATA IS REQUIRED.

LIST III

Response Time Analysis



APP/LOC.= CICSLA AVG RESP TIME= 0:00:03 DATE = 19MAY87
310 TRANS.<1 2009 TRANS.<2 1903 TRANS.<3 1717 TRANS.<5 677 TRANS.>=5

Rohm and Haas Company. GRADAY

TABLE 1

PEAK HOURS RESPONSE TIME STATISTICS BY PERFORMANCE CLASS

OBS	NSVPCLSS	NSVTRESC	NSVRDST1	NSVRDST2	NSVRDST3	NSVRDST4	NSVRDST5	NSVPCTR1	NSVPCTR2	NSVPCTR3	NSVPCTR4	NSVAVTTM
1	CICSCH	9066	6474	1777	381	269	165	71.40 %	91.00 %	95.21 %	98.17 %	0:00:01.19
2	CICSLA	2259	96	630	655	595	283	4.24 %	32.13 %	61.13 %	87.46 %	0:00:03.54
3	CICSLCL	16175	12586	2370	614	339	266	77.80 %	92.46 %	96.25 %	98.35 %	0:00:01.13
4	DFLTLCL	141	12	36	27	29	37	8.51 %	34.03 %	53.19 %	73.75 %	0:00:04.66
5	DFLTRMT	123	29	34	27	25	8	23.57 %	51.21 %	73.16 %	93.49 %	0:00:02.47
6	IDMSENG	20	6	11	3	0	0	30.00 %	85.00 %	100.00 %	100.00 %	0:00:01.34
7	IDMSLCL	293	139	66	21	23	44	47.43 %	69.96 %	77.13 %	84.98 %	0:00:05.18
8	TSOCH	35	21	1	1	2	10	59.99 %	62.85 %	65.71 %	71.42 %	0:00:05.21
9	TSOENG	4502	3544	507	144	116	191	78.71 %	89.98 %	93.17 %	95.75 %	0:00:01.42
10	TSOLCL	14873	11084	1883	682	519	705	74.52 %	87.18 %	91.76 %	95.25 %	0:00:01.56
11	TSOSH	5385	3388	250	169	612	966	62.91 %	67.55 %	70.69 %	82.05 %	0:00:02.11
12	TSOVAL	81	5	14	13	21	28	6.17 %	23.45 %	39.50 %	65.42 %	0:00:07.00
13	VMCH	123	78	33	6	1	5	63.41 %	90.24 %	95.12 %	95.93 %	0:00:01.20
14	VMENG	150	125	20	4	1	0	83.33 %	96.66 %	99.33 %	99.99 %	0:00:00.80
15	VMLCL	398	288	73	16	10	11	72.35 %	90.70 %	94.72 %	97.23 %	0:00:01.16
16	VMSH	528	303	27	29	105	64	57.38 %	62.49 %	67.99 %	87.87 %	0:00:01.26
17	VMVAL	25	4	12	5	4	0	16.00 %	64.00 %	84.00 %	100.00 %	0:00:01.98

TABLE 2

Local:

OBS	TSO TERMINAL ID	AVG RESPONSE TIME ALL FUNCTIONS	NETWORK DELAY	TSOMON REPORTED END-USER RESPONSE	NLDM REPORTED END-USER RESPONSE
1	A001TD17	0:00:00.50	0:00:00.06	0:00:00.56	0:00:00.72
2	A001TD20	0:00:01.57	0:00:00.09	0:00:01.66	0:00:01.57
3	A001TD31	0:00:00.82	0:00:00.06	0:00:00.88	0:00:01.10
4	A001TE00	0:00:01.13	0:00:00.06	0:00:01.18	0:00:02.22
5	A001TE19	0:00:01.35	0:00:00.13	0:00:01.48	0:00:01.73
6	A001TE31	0:00:03.35	0:00:00.18	0:00:03.53	0:00:15.89
7	A001TF05	0:00:00.82	0:00:00.06	0:00:00.89	0:00:00.84
8	A001TF06	0:00:01.76	0:00:00.16	0:00:01.92	0:00:01.29
9	A001TF13	0:00:39.97	0:00:00.15	0:00:40.12	0:00:47.52
10	A001TF16	0:00:04.16	0:00:00.80	0:00:04.96	0:00:05.01
11	A001TF20	0:00:05.12	0:00:03.35	0:00:08.47	0:00:05.94
12	A001TF21	0:00:06.31	0:00:00.34	0:00:06.65	0:00:08.57
13	A001TF23	0:00:04.84	0:00:00.05	0:00:04.88	0:00:03.01
14	A001TF26	0:00:01.76	0:00:00.05	0:00:01.81	0:00:02.19
15	A001TF27	0:00:02.00	0:00:00.62	0:00:02.62	0:00:05.28

Remote:

OBS	TSO TERMINAL ID	AVG RESPONSE TIME ALL FUNCTIONS	NETWORK DELAY	TSOMON REPORTED END-USER RESPONSE	NLDM REPORTED END-USER RESPONSE
1	B009TC20	0:00:02.45	0:00:00.30	0:00:02.74	0:00:01.62
2	B009TC21	0:00:02.43	0:00:00.30	0:00:02.73	0:00:03.93
3	B009TC27	0:00:02.83	0:00:00.87	0:00:03.69	0:00:02.40
4	B009TD18	0:00:10.12	0:00:00.33	0:00:10.45	0:00:10.66
5	B009TD47	0:00:03.98	0:00:00.36	0:00:04.34	0:00:05.18
6	B009T000	0:00:05.01	0:00:00.44	0:00:05.45	0:00:10.12
7	B009TS00	0:00:02.22	0:00:02.15	0:00:04.37	0:00:02.22
8	B009TS16	0:00:04.93	0:00:01.96	0:00:06.89	0:00:16.54
9	B009TS32	0:00:03.03	0:00:00.43	0:00:03.46	0:00:04.10
10	B009T103	0:00:04.49	0:00:01.46	0:00:05.94	0:00:04.75
11	B009T132	0:00:00.78	0:00:00.35	0:00:01.13	0:00:01.27
12	C009TA23	0:00:00.91	0:00:00.39	0:00:01.30	0:00:01.29
13	C009TA25	0:00:02.38	0:00:08.06	0:00:10.44	0:00:03.72
14	C009TA26	0:00:04.74	0:00:00.42	0:00:05.16	0:00:05.26
15	C009TA27	0:00:01.99	0:00:00.70	0:00:02.69	0:00:03.32

GENERAL PRINCIPLES FOR PERFORMANCE ORIENTED DESIGN

Connie U. Smith
Performance Engineering Services
L&S Computer Technology, Inc.
P.O. Box 9802 - M.S. 120
Austin TX 78766
(505) 988-3811

ABSTRACT

Seven principles for designing responsive systems are presented. They generalize and formalize the expert knowledge that performance specialists use to build or tune systems. The principles are oriented to large systems of programs in early development -- requirements definition and preliminary design creation. Most software designers can easily understand and use the principles to build better systems. The paper introduces and illustrates the principles, discusses their effective use, and suggests how they can be useful to a performance specialist who already possesses expert knowledge.

1.0 INTRODUCTION

Software Performance Engineering (SPE) calls for building performance into new systems rather than tuning it in after implementation. It prescribes models for predicting software performance and a software engineering methodology for gathering data and interfacing the SPE activities with the software engineering process [SMI81, SMI84]. If you have tried it, you have probably discovered that it is possible to detect performance problems early in the development process while there is still time to correct them. You have probably also found that it is easy to detect performance problems in software designs -- they are "obvious" to performance specialists -- yet it is often necessary to construct performance models to prove it. Even then, designers may require justification of the model assumptions rather than immediately recognize and correct software problems.

In applying SPE to numerous software development projects, we have observed much commonality in the design problems and gained some insight into why they occur. This paper presents some general principles for performance oriented design that most designers can easily understand and use to build better systems. To experienced performance analysts they are not revolutionary new ideas. They are, however, a generalization and abstraction of the "expert knowledge" that performance specialists use in building or tuning systems. They are also an effective way to communicate this knowledge to software architects and designers.

As mentioned, the SPE methodology and models have been around for a while [BOO79, SAN78, SAN79, SHO75, SMI80a, SMI80b]. Similarly, program efficiency guidelines are well known [BEN83, FER78, FER83, KNU68, KNU71, KNU73]. Alternatively, these general principles apply to large ***systems*** of programs in ***early*** life cycle stages, when one is concerned with formulating requirements and design specifications that will lead to ***systems*** with acceptable ***responsiveness***. While many of the fundamental performance concepts are similar, large system design is different. Tuning transforms a program into a more efficient program that performs the same function. The principles address ***what*** functions a system performs as well as ***how*** they are performed. Similarly, responsiveness to users is not necessarily the same as efficient use of computer resources. Note that the principles ***supplement*** the SPE models, they do not replace them.

The next section introduces the seven principles, and Section 3 discusses using them effectively. Then Section 4 suggests how they are useful to a performance specialist who is already familiar with the performance concepts.

2.0 SYNTHESIS PRINCIPLES

These principles have been developed through practical experience with large software systems. On one particular project, severe performance problems were detected during the system integration phase. A thorough performance "tuning" study was conducted that resulted in numerous proposals for improvements. Many were deemed infeasible because of the magnitude of the change at the late development stage. Others were implemented as proposed. Each of the proposed improvements was catalogued and classified by the type of change. The classifications were related to a performance model such as that in the Appendix, and the generalized set of principles was formulated on the basis of the relationship between the changes and the impact on the parameters of the performance models.

The principles apply to software systems, large or small, executing on a spectrum of computer systems: microcomputers, large mainframe computers, distributed systems, and MIMD computers.[1] They are most effective for very large software systems with high processing demands.

The following sections present the seven general principles for the synthesis of responsive software systems: fixing, locality design, processing versus frequency tradeoff, shared resources, parallel processing, centering, and instrumenting. Each principle is defined and explained using simple examples. A hypothetical automated teller machine (ATM) example illustrates many of the principles. Examples are provided for both system requirements and design tradeoffs.

2.1 FIXING POINT PRINCIPLE

Fixing is the mapping of an action or function desired to the instructions that accomplish the action. It is also the mapping of information required to the data used to produce it. The fixing "point" is a point in time. The latest fixing point is during execution immediately before the action or information is required. Fixing could be done at several earlier points: earlier in the execution, at program initiation time, at compilation time, or external to the software system.

> ***Fixing Point Principle: for responsiveness, fixing should be done at the earliest feasible point in time such that retaining the fixing information is cost-effective.***

This is often viewed as a resource usage versus flexibility trade-off. Early fixing leads to far more responsive systems while later fixing is more flexible and general. Late fixing provides flexibility in two ways: through automatic linking of actions (or information) to the most recent version of the corresponding instructions (or data), and by imposing fewer constraints on which actions (or information) can be executed (or accessed). Late fixing of actions affects responsiveness since the code for fixing the actions to the instructions must be executed every time the actions are requested. Similarly, late fixing of information to data requires executing the fixing code every time the information is accessed. The fixing code is executed unnecessarily when no changes have occurred since the last use and when common, predictable actions and information are used.

1. When the processors have conventional von Neuman architectures. This excludes pipelined processors, systolic arrays, and other special purpose architectures. Such computer systems may have additional performance-determining factors not addressed here.

Often, software is more general than required. When users are uncertain how they will use the system, they request generality. However, it is better to consider early its effect on performance, to invest the time to study how systems will be used, and to include generality only where it is really needed. As a compromise, early fixing should be done for the functions that are used most frequently, and generality included only for the less frequent functions.

It is cost-effective to retain the fixing information when the cost of retaining it is less than the cost of fixing multiplied by the probability that re-fixing is unnecessary. Jobs or transactions will be more responsive when customized interfaces are designed with early fixing for common predictable actions and information, and when baselined versions of information are changed infrequently and are fixed early. For flexibility, special interfaces can be provided and used only when needed for uncommon actions, for information infrequently accessed, and for re-fixing to more recent versions as required.

An example of fixing requirements is choosing the interface for DBMS information selection. Runtime fixing is done when general ad-hoc queries against the information are allowed, and the query is parsed and satisfied at runtime. Alternatively, the information can be fixed earlier, at compile time, with managed queries: pre-defining the information that is to be quickly accessible and building a menu with only those items. The data is still selected at run time, but the code to retrieve it is fixed at compile time.

Note that binding is a subset of fixing. An example of fixing that is not binding is determining when data in files or internal tables should be sorted. Files or tables kept in the desired order, with all additions preserving the order, are fixed early. Late fixing applies to ordering them when needed. (Some experts would consider this an example of "extended binding." Unfortunately, many subconsciously view binding in a limited context, so the term *fixing* is used to encourage broader interpretation.)

2.2 LOCALITY DESIGN PRINCIPLE

Locality refers to the closeness of the mapping of logical tasks and resources to physical resources. According to Webster, ***close*** means "being near in **time, space, effect** (that is, purpose or intent), or **degree** (that is, intensity or extent)."

The dictionary specification for close mapping then leads to four types of locality design for performance engineering. They are illustrated in the following example. Consider the logical task to sort a list of integers. *Temporal locality* of the mapping of this logical task to the physical resources is better if the integers are all mapped at the same time to the physical processor that sorts rather than one at a time (with a large interval of time between each). Similarly, *spatial locality* is better if the task is in a location that is near the physical resource, such as in memory that is directly accessible by (local to) the processor on which it will execute, rather than located on a disk drive attached to a different machine. The task could be mapped to different types of physical processors; consider the choice of mapping to a general purpose CPU or mapping to a special purpose chip designed specifically to sort lists of integers. *Effectual locality* is better for the special purpose processor since its purpose matches the task more closely than the general purpose CPU. *Degree locality* refers to the extent of the task, as in the length of the list of integers as compared to the size of the processors (e.g., speed, memory, size, etc.).

The locality design principle can thus be stated as follows:

> ***Locality Design Principle: Systems should be designed to have a close mapping of logical tasks and resources to physical resources.***

Spatial locality is in menu networks that cluster related activities into a single menu and changes menus when user activity changes. For temporal locality, those activities that are requested most frequently should be in the first menu seen by the user; infrequent activities should be seen much later. (Since they are clustered into the same menu, they also have good spatial locality; temporal locality refers to *when* they appear in the sequence of menus.) Effectual locality is found in most ATM's: they use a special purpose automated teller machine that has a user interface customized to the ATM application, rather than a general purpose terminal console. Degree locality matches the size of the ATM application to the machine. A microprocessor is sufficient, a supercomputer is not needed.

2.3 PROCESSING VERSUS FREQUENCY TRADEOFF PRINCIPLE

This principle addresses the amount of work done for each processing request, and its impact on the number of requests made. The "tradeoff" concerns the cases when more work per request reduces the number of requests made, and vice versa. The principle is as follows:

> ***Processing versus Frequency Tradeoff Principle: Minimize the processing times frequency product.***

A requirement tradeoff for the ATM example is determining whether multiple transactions per ATM session are to be allowed. The prompt for continuing a session requires some additional processing time, but the total number of sessions may be less than when a separate session is required for each transaction.

Another example is in displaying results of a database query when multiple data items are selected. Either all results can be displayed with a GETALL, or the first item can be displayed with a GETFIRST and each additional one with a GETNEXT. If users frequently wish to display all results, then the GETALL command may be desired for convenience. The principle can then be applied to the design. A design with a direct GETALL has a high bandwidth interface to the database; a design with an intermediate level of abstraction intercepts the user's GETALL request, issues multiple calls to the database (the GETFIRST and multiple GETNEXT's), accumulates the results, and transmits all back to the user at once.

There is not always a tradeoff in the two factors. For example, for file I/O one can retrieve one byte of information at the same relative cost as 20 bytes of information, because the processing time is dominated by the I/O operation. Therefore, if most of the time the additional information is eventually required, there will be essentially no change in processing time *and* the number of requests will be reduced. Thus there is no processing time penalty for reducing the number of requests.

There are three special cases of this principle that commonly occur in large software systems. The special cases relate more directly to decisions made during software development and should provide a better understanding of the general principle. In the following paragraphs each of the special cases is defined and an example is included. The first two cases illustrate increasing processing and reducing frequency. The third illustrates decreasing processing.

The first is the "family plan." The analogy is providing service for the first family member for full fee, and thereafter providing services for all additional family members at the same time at a negligible additional fee. An application is an "access list" strategy for maintaining a list of resources that a user has permission to access with the types of access allowed for each. A "security guard" within an operating system can obtain the access rights for all resources of a particular user at a negligible additional cost over obtaining the access rights for one resource. The family plan applies when multiple requests for resources from the same user are likely to occur within a short period of time.

A variation is the "maximum use of overhead costs." With it, service requests are grouped into "batches" to process multiple requests for each overhead penalty. An application is a hypothetical distributed computing environment where many users share a local machine for editing, compiling, and executing minor tests. Major tests are submitted to a remote machine. Submission requires the execution of "overhead" software to establish the communication link, to transmit the request, and to receive the reply. The "maximum use of overhead costs" implies that requests for major tests should be collected (delayed) and periodically transmitted to the remote machine. It is applicable when the batching delay is small compared to the duration of the major test, and it is likely that multiple requests will be made during that delay period.

The third special case is the "expensive processing reduction." With it, expensive processing is done only when it is necessary. An application is a database with a high degree of locality of access to the data. With "expensive processing reduction," when information is requested, the software first checks to see if it has been recently retrieved. It is applicable when it is highly probable that the check will be successful, and the processing time is substantial. Another example is in reservation systems; when a customer calls to check previously made airline or car rental reservations, a short record that contains frequently accessed information is often fetched first. Processing takes longer for less frequent requests such as reserving a special meal.

2.4 SHARED RESOURCE PRINCIPLE

Computer system resources are limited and must be shared. To share, either multiple processes can use the resource at the same time, or they can take turns, each process using the resource one at a time (multiplexing). The management of shared resources affects the software in two ways: the additional processing overhead for scheduling the resource, and the possible additional time for waiting to gain access to the resource. The general principle is as follows:

> ***Shared Resource Principle: resources should be shared when possible. When exclusive access is needed, the sum of the holding time and the scheduling time should be minimized.***

By sharing resources (rather than taking turns), the overhead for scheduling is minimized and there is minimal wait to gain access (there may be a wait if another process already has exclusive access even though the requestor is willing to share).

Decreasing the sum of the holding time and the scheduling time of multiplexed resources decreases the *average* wait time to gain access to the resource.[2] There are four ways to minimize the holding time:

1. *Minimize the processing time (using the other principles)*
2. *Hold only while needed*
3. *Request smaller resource units*
4. *Fragment the resource requests*

The first decreases the holding time by doing less work while the resource is held. The second says that a multiplexed resource should be requested just before it is used and released immediately afterwards. The third, requesting smaller resource units, means that less of the resource is held. The fourth, fragmenting requests, means partitioning one request that requires a long holding time into shorter requests, each of which requires a shorter holding time.

Minimizing the holding time of a multiplexed resource may increase scheduling time. The scheduling of service requests requires more processing for smaller resource units; and fragmenting requests into multiple shorter requests introduces more requests for scheduling service. ***There will be a net improvement only when the additional processing is less than the expected wait time using the larger units or longer requests.***

In the ATM example, when a function is requested that changes the account master file, such as a withdrawal, only one request can be allowed to modify the file at a time to ensure file integrity. This can be implemented by allowing only one ATM to be active at a time. That is, the central computer completes one ATM transaction before beginning the next. An alternative is to allow multiple ATM's to be active concurrently and to add additional processing to request exclusive access to the account master file only during the critical time when the balance is being changed. The preferable alternative and the magnitude of the savings depend on the frequency of requests that require exclusive access to the master file.

The above was an example of locking only when needed. Smaller resource units implies locking at an account level rather than for the entire database. Locking for the entire database requires only one lock, while locking for accounts requires many. Therefore, the scheduling time to search through the list of locked accounts is greater than the time to check the flag for the entire database. Of course, the space requirements are greater as well.

There are also several choices for the degree of fragmentation requests. The software could lock for the first request that requires a change to the database and hold it until the customer completes the ATM session. Alternatively, it could lock and free the database immediately before and after any change.

2.5 PARALLEL PROCESSING PRINCIPLE

Processing time can sometimes be reduced by partitioning a process into multiple concurrent processes. The concurrency can either be real concurrency where the processes execute at the same time on different processors, or it can be apparent concurrency where the processes are multiplexed on a single processor. For real concurrency, the processing time is reduced by an amount proportional to the number of processors. Apparent concurrency is more complicated. Although some of the processing may be overlapped (e.g., the CPU, memory, or files), additional wait time may be introduced. Additional overhead processing may also be required to manage the communication and coordination between concurrent processes.

The principle is as follows:

> ***Parallel Processing Principle: parallel processing should be exploited (only) when the processing speedup offsets the communication overhead and the resource contention delays.***

In the ATM example, the microcomputer at the ATM site allows for parallel processing. The following is one such strategy:

- The ATM micro computer requests the account information and the authorization code, and sends it to the central computer.
- While the central computer is verifying the account number and authorization code, the ATM microcomputer obtains the transaction selection and information required.
- It formats and sends the transaction information to the central computer. While the transaction is being processed at the central site, the ATM begins printing the receipt for the customer.
- While waiting for the response, the ATM micro asks if another transaction is desired, and if so, begins processing it.

2. The shared resource principle is a synergistic principle; it improves the average waiting time for all processes, rather than improving a process's own responsiveness. See Appendix for further information.

This processing strategy is complex since many special cases must be handled. For example, the customer might first request the account balance; while that transaction is being processed by the central computer, the customer requests a withdrawal. When the account balance arrives from the central computer, the balance is insufficient to cover the withdrawal. This does not introduce a processing time delay, but it complicates the number of situations that the microcomputer software must consider.

In general, the benefits derived through apparent concurrency are not significant compared to those achievable using the other principles. It has the further disadvantage of adding complexity to the software system. Real concurrency will be effective if the processing time reduction is much greater than the additional overhead for communication and coordination of the concurrent processes. The performance improvement must also be weighed against the cost of the additional processing power and the cost of more complex software, to determine whether it will be effective.

2.6 CENTERING PRINCIPLE

The five previous principles provide guidance for the synthesis of software requirements and designs. Their application improves the performance of the part of the system to which they are applied. This principle is different in that it addresses leveraging the application of the principles by focusing attention on those portions of a large software system that have the greatest impact on its responsiveness. It is based on the folkloric "80-20 rule" for the execution of code within programs. A generalized version is: a subset ($\leq$ 20%) of the *system functions* that are available will be requested by the system users most ($\geq$ 80%) of the time. These frequently requested functions are called the dominant workload functions (they also cause a subset ($\leq$ 20%) of the *programs or modules* in the software system to be executed most ($\geq$ 80%) of the time, and the code within modules, etc.) Performance enhancements made to these key areas of the software system thus greatly affect the overall responsiveness of the system.

The principle is as follows:

> ***Centering principle: identify the dominant workload functions and minimize their processing.***

That is, create special execution paths for the dominant workload functions that are customized and trimmed to include only processing that *must* be done while the user waits. The principles in Sections 2.1 through 2.5 are applied to the special paths to minimize their processing. Separate transactions should be constructed for the workload functions that are requested less frequently.

Most automated teller machines apply the centering principle to system requirements by including a "quick withdrawal" transaction that reduces processing by eliminating prompts and processing for account type, amount, and additional transactions; and (on some ATM's) by omitting the new account balance from the receipt. They also apply it to the system design by customizing the access methods for the small percentage of customers who are likely to use ATM's.

Note that centering emphasizes optimizing frequent requests. Although it is not explicit, this implies that is generally not profitable to worry about the responsiveness of functions that are infrequently used; the modifications may lead to negligible improvements, or even to degradations. The exceptions are for important, but infrequent, functions or for functions that have large processing times.

The centering principle applies to all systems, but *what* one centers on may depend on the performance goal or the type of system. Thus far, the discussion has implicitly addressed application software systems that support online interactive users where the performance goal is responsiveness to users. When system throughput (number of responses per unit time) is a performance goal, or general purpose software systems do not have a dominant workload, centering is on those components with the largest cumulative space - time product across the specified usage scenarios. Note that *most* general purpose systems (e.g., database systems, operating systems, or other commercial products) have dominant workloads. The developers may not know what they are, but they can be identified and it is important to do so. If one instead focuses only on the components with a large cumulative space - time product, improvements can be made by reducing their time, but other opportunities to shorten path lengths by eliminating general processing steps not applicable to the dominant workload may not be found.

Early in the life cycle, centering focuses on the functions frequently requested by users. Later, during implementation, the centering principle also addresses software components with large resource demands, the "major consumers," even though they may not be executed frequently. Because the major consumers hold resources, they may cause excessive delays to the dominant functions. While they can be identified earlier, they are typically less important than the dominant functions. They are addressed later, but do not drive the design earlier as do the frequent functions.

There is an additional difference in the early and late life cycle centering considerations: the effect of the improvements. Early life cycle centering focuses on the functions frequently requested by the users. We assume that reducing the processing time for these requests does not affect the number of times they are requested. Later in the life cycle we also address the major consumers of resources. When their processing time is reduced, the components that fall in the set of major consumers of resources may change. Thus, reducing the resource requirements of the major consumers could be an endless process, since there will always be major consumers of each resource. It is not endless because we focus on achieving the performance goal, not on minimizing resource usage.

2.7 INSTRUMENTING PRINCIPLE

Instrumenting software means inserting code at key probe points to enable the measurement of pertinent execution characteristics. The principle is:

> ***Instrumenting Principle: Systems should be instrumented to enable measurement and analysis of workload scenarios, resource requirements, and performance goal assessments.***

This principle does not in itself improve performance, but is essential to **controlling** performance. It has its foundations in engineering, particularly process control engineering. Their rule of thumb is "if you can't measure it, you can't control it."[3]

Probes should be inserted to collect the number of times workload scenarios are executed, how often software components are executed, and their resource requirements. Performance engineering also calls for identifying the system functions that are used, and the information that is accessed. Since the functions are fixed to software components and information is fixed to data that is accessed, instrumentation is needed to identify and measure the relationships. This discussion focuses on instrumentation for performance engineering; data should also be collected to control other aspects of software quality.

For the ATM example, instrumentation is needed to enable collection of:

- Number of times each ATM site is used, customer arrival rates, and the peak load characteristics
- Number of times each ATM function is selected, and the number of times errors are made

3. I have always believed that instrumenting software is important to performance. Nevertheless, I omitted instrumenting in my original set of principles since it does not directly improve performance. Kopetz presented a paper on principles for designing fault tolerant real-time process control systems, and included an instrumenting principle [KOP86]. He persuaded me that it is vital and should be adopted, even though it does not directly improve performance.

- Number of times each software component is executed
- For each component: the amount of CPU time, number of I/O's, memory, procedure calls, etc.
- The withdrawal amounts, the accounts accessed, the number of transactions local to the bank versus those that access a national network, and the number of transactions in a session
- The total time for: the user to respond to requests for information, the completion of a transaction, the completion of a session

The above is a partial list of the data of interest. Collection of this data should be considered when the system requirements and design are conceived; it is much more difficult to obtain later. This is because of limitations in instrumenting technology - most tools collect system level data such as program execution time rather than functional data such as end to end response time (e.g., ATM transaction or session time). Functional data currently requires programmers to insert code to call system timing routines and write the key events and pertinent data to files for later analysis. Additional tools and techniques are needed to automate the data collection and facilitate the data analysis.

2.8 SUMMARY

In this section, the principles for synthesis of responsive software systems were introduced and illustrated with simple examples. Most of them involve tradeoffs. A quantitative analysis of the performance effect of the fixing point, processing versus frequency tradeoff, and centering principles is in [SMI86b]. While they can be evaluated with simple back-of-the-envelope calculations, the others require more sophisticated performance models. Since the performance engineering methodology in [SMI81] incorporates the modeling activity into the software design process, it is easy to use the models to quantify the tradeoffs. Many performance analysts have the necessary modeling skills, and numerous support tools are available.

3.0 STRATEGY FOR EFFECTIVE OPTIMIZATION

The general principles are necessary for developing systems with good performance characteristics, that is, for "doing it right the first time," but they are not sufficient. There are four factors that determine their effectiveness; the principles must be:

1. *Applied to appropriate software components*
2. *Necessary*
3. *Lead to global improvement*
4. *Cost-effective*

Each of these is explained in the following paragraphs.

The principles must be applied to the components that are critical to performance. Identifying them at design time can be difficult; and intuition can be misleading. Some designers mistake the components that are most difficult to design and implement for the critical components; whereas, the critical components are generally those most frequently executed.

For example, network security management is difficult to design because many complex situations must be handled. Security may require a fair amount of execution time; however, other network software components, such as the communication protocol routines, are generally more frequent. Security management *may be* critical to performance, modeling determines its impact.

The second effectiveness factor is applying the optimization efforts to software components only when *necessary*. Usually, it is unnecessary to overachieve a performance goal; therefore, if the performance goal can be easily attained, it is not effective to devote valuable development time to extensive performance enhancements. Similarly, it is not effective to expend much effort optimizing components of the software system that have little impact on overall performance.

Another aspect of necessary optimizations is distinguishing software requirements that are necessary from those that are both unnecessary and adversely affect responsiveness. Sometimes these "artificial" requirements are introduced at design time with the intent to improve performance. An example is (an extra) requirement for a component that produces a list of items: that the list be in sorted order. The order requirement may be artificial, the intent being to reduce subsequent search time. Performance models show whether the sort improves overall responsiveness. A straightforward approach (eliminating the sort) may not degrade responsiveness, and will result in less code to maintain.

The third effectiveness factor says that optimization techniques should result in *global improvements* to the software performance. This is most important in detailed design and coding stages, when many people are involved in development. Optimizations made in one part of the software system must be consistent with those made in other areas. Performance models quantify the overall effects.

The last of the four factors is that the performance improvement must be *cost-effective*. The time to implement the optimizations must be weighed against expected savings. It may be extremely difficult to achieve a specified performance goal for a large software system. Achieving the goal may be possible only at great costs in personnel time and in elapsed time to implement the system. It is important to estimate the cost of achieving the performance goal and insure that it is justified. It is often possible to negotiate for more reasonable performance goals before expending excessive efforts to achieve unrealistic ones.

4.0 RELEVANCE TO PERFORMANCE SPECIALISTS

These principles formally state concepts that are already familiar to experts in performance oriented design, so why is this paper appropriate for CMG audiences? Suggestions on how to use these principles to improve software performance in your organization follow.

The first and easiest use is to incorporate the principles into software performance engineering studies. SPE calls for systematically assessing performance trade-offs throughout development. The principles guide the identification of improvements and help explain the rationale for the changes to designers. The instrumenting principle suggests working with designers to help them identify what should be measured, and the best way to do so. It also suggests developing standard software for measurement data analysis and reporting that can be used by all new software systems.

There are obvious differences in personal aptitudes for performance oriented design. While some designers will never be interested in performance, many others are interested in and willing to create high performance systems, but they lack the necessary training and experience. These people quickly grasp the principles, especially when they see examples relevant to their applications. Another use for the principles is to educate designers. Publicize references and suggested reading material in internal newsletters. Suggest LAM84, SMI86a, SMI87, or KOP86 (for real-time process control systems). Suggest commercial seminars on software performance engineering to your designers, or develop your own. Develop some applications of the principles using your systems as examples. Many important design issues are discussed in SMI87, such as communication, human interface design, data organization, and information hiding. Customize those examples to your environment and educate your designers.

Another way to use the principles is to create software tools to help designers evaluate trade-offs. Figure 4.1 shows a hypothetical tool to evaluate the effect of centering by creating special processing paths for dominant workload functions. It asks for estimates of processing and frequency, and produces the resulting average processing time and time savings. Your tools could also interface with software model analysis tools to calculate the processing time, and commercial system modeling tools to solve for response times. The formulas that provide these results are in [SMI86b].

You could also produce some "engineering reference tables" using the formulas in [SMI86b]. Figure 4.2 shows a table for evaluating the effect of centering. For example, if the general path takes 5 seconds, a special path could be created that takes 1 second, and 9 out of every 10 requests are for the special path, a designer could compare the average processing time with the special path (1.4 seconds) to the average time without it (5 seconds). You can generate a handbook of tables like this for parameter values that are common in your environment and for the other general principles. You could also estimate the effect on response time (rather than on processing time) using estimates of queueing delays for your environment.

Another use for the principles that appears to be viable in the near future is an expert system for performance oriented design. By formalizing the expert knowledge, we are one step closer to automating the detection and correction of performance problems.

Figure 4.1
Hypothetical Trade-off Analysis Tool

Input:	
Name of special function	*quick withdrawal*
Additional processing to identify it (seconds)	*0*
Processing for general path (seconds)	*20.6*
Processing for (separate) function (seconds)	*6.8*
Ratio of special to total functions	*9/10*
Report:	
Processing time - general path (seconds)	*20.6*
Processing time - special function (seconds)	*6.8*
Average processing time (seconds)	*8.18*
Average processing time savings (seconds)	*12.42*

Figure 4.2
Engineering Reference Table: Centering

Average Processing Time

Probability of special path = .09
Additional processing to identify special path = 0

Processing for special path:

	0.1	0.5	1	1.5	2	2.5	3	3.5	4
Processing for general path:									
1	0.19	0.55	1	1.45	1.9	2.35	2.8	3.25	3.7
2	0.29	0.65	1.1	1.55	2	2.45	2.9	3.35	3.8
3	0.39	0.75	1.2	1.65	2.1	2.55	3	3.45	3.9
4	0.49	0.85	1.3	1.75	2.2	2.65	3.1	3.55	4
5	0.59	0.95	1.4	1.85	2.3	2.75	3.2	3.65	4.1

References

[BEN83] Bentley, J.L. *Writing Efficient Programs*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[BEN84] Bentley, J.L, "Programming Pearls," *CACM*, 27, 11, November 1984, 1087-1092.

[BOO79] Booth, T.L., "Use of computation structure models to measure computation performance," *Proc. Conf. on Simulation Measurement and Modeling of Computer Systems* , Boulder, Aug. 1979, 183-188.

[FER78] Ferrari, D. *Computer Systems Performance Evaluation.* Prentice-Hall, Englewood Cliffs, NJ , 1978.

[FER83] Ferrari, D., Serazzi, G., Zeigner, A. *Measurement and Tuning of Computer Systems.* Prentice-Hall, Englewood Cliffs, NJ , 1983.

[GRA78] Graham, R.M., Clancy, G.J., DeVaney, D.B., "A software design and evaluation system," *CACM* 16, 2, Feb. 1973, 110-116.

[KNU68] Knuth, D.E. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms.* Addison-Wesley, Reading, MA , 1968.

[KNU71] Knuth, D.E. An empirical study of FORTRAN programs. *Software Practice and Experience*, 1, 2, Apr. 1971, 105-133.

[KNU73] Knuth, D.E. *The Art of Computer Programming, Vol. 3: Sorting and Searching.* Addison-Wesley, Reading, MA, 1973.

[KOP86] Kopetz, H., " Design principles for fault tolerant real time systems," *Proc. Hawaii International Conference on System Sciences* 19, Jan. 1986, 53-62.

[LAM84] Lampson, B.W., "Hints for computer system design," *IEEE Software*, Feb. 1984, 11-28.

[LAZ84] Lazowska, E.D., Zahorjan, J., Graham, G.S., Sevcik, K.C., *Quantitative System Performance.* Prentice-Hall, Englewood Cliffs, NJ , 1984.

[SAN78] Sanguinetti, J.W., "A formal technique for analyzing the performance of complex systems," *Proc. Computer Performance Evaluation Users Group 14*, Boston, Oct. 1978, 67-82.

[SAN79] Sanguinetti, J.W., "A technique for integrating simulation and system design," *Proc. Conf. on Simulation Measurement and Modeling of Computer Systems* , Boulder, Aug. 1979, 163-172.

[SAU81] Sauer, C.H., Chandy, K.M., "*Computer Systems Performance Modeling*," Prentice-Hall, Englewood Cliffs, NJ , 1981.

[SHO75] Sholl, H.A., Booth, T.L., "Software performance modeling using computation structures," *IEEE TSE* 1, 4, Dec. 1975.

[SMI80a] Smith, Connie U., Browne, J.C., "Aspects of software design analysis: concurrency and blocking," *Proc. Performance 80* , Toronto, May 1980, 245-254.

[SMI80b] Smith, Connie U., The prediction and evaluation of the performance of software from extended design specifications, Ph.D. Dissertation, U. Texas at Austin, University Microfilms Pub. #KRA81-00963, 1980.

[SMI81] Smith, Connie U., "Software performance engineering," *Proc. Computer Measurement Group Intl. Conf. XIII* , New Orleans, Dec. 1981, 5-14.

[SMI85] Smith, Connie U., "Performance engineering: A bibliography," Special issue on Software Performance Engineering, *Computer Measurement Group Transactions*, 49, Sept. 1985, 63-68.

[SMI86a] Smith, Connie U., "Synthesis principles for high performance software," *Proc. Hawaii Intl. Conference on System Science* 19, Jan. 1986, 17-27.

[SMI86b] Smith, Connie U., "Independent General Principles for Constructing Responsive Software Systems," *ACM Transactions on Computer Systems*, 4, 1, Feb. 1986, 1-31.

[SMI87] Smith, Connie U., "Applying Synthesis Principles to Create Responsive Software Systems," *IEEE TSE*, to appear.

Appendix
Basis for the Principles

The computer system on which the software executes can be viewed as an abstract model as in Figure A.1. Figure A.1a shows several types of jobs (A,B,C,...) arriving for service, possibly waiting in a queue for their turn for service, then leaving upon completion. Figure A.1b shows an expanded model that identifies computer system resources that each of the jobs may use while being served. It is well known that the performance of such a system depends on the following parameters of the model [LAZ84, SAU81]:

- the arrival rate of each type of job
- the computer system resource requirements of each type of job
- the contention delays that result from the interaction with other jobs in the system
- the scheduling policies used to determine which waiting job next obtains the needed computer system resource

The seven principles improve performance by favorably affecting the corresponding system performance parameter. Table A.1 summarizes the correspondence:

Table A.1

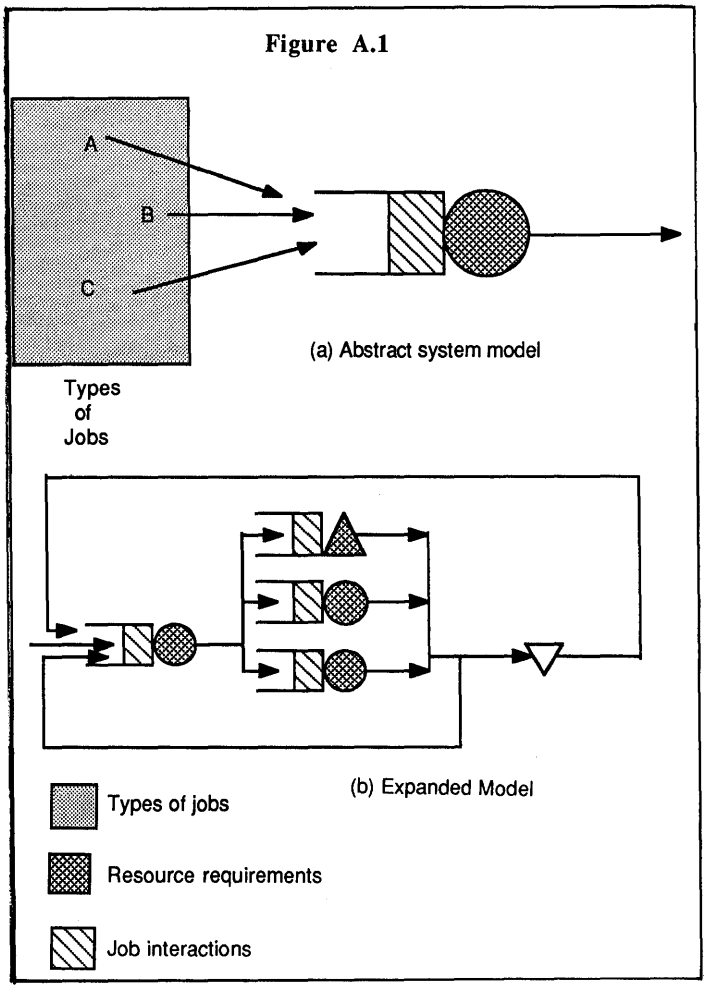
System Parameter	Performance Principle	Type
Resource Requirements	Fixing Point	Independent
	Locality Design	Independent
	Processing vs. Frequency Tradeoff	Independent
Job Interactions	Locality Design	Synergistic
	Shared Resources	Synergistic
	Parallel Processing	Synergistic
Types of jobs	Centering	Independent
Scheduling	—	—
—	Instrumenting	Control

The fixing point principle concentrates on when and how often processing occurs. Locality design pertains to effective use of resources. Processing versus frequency tradeoff affects the number of requests for resources and the amount requested. Shared time interval. The extent of parallelism and the competition for resources among the parallel tasks is affected by the parallel processing principle. Centering focuses on the jobs that are key to the responsiveness of the system. The scheduling of jobs is not addressed because it is generally a service provided by the computer system, and not frequently a decision that must be made in the early development stages of new software systems. If scheduling is addressed, it is usually only after implementation when job priorities are manipulated. The instrumenting principle does not directly influence the performance-determining factors. Instead, it addresses ***controlling*** the other factors.

There are three types of principles shown in the table, independent, synergistic and control. Independent principles improve the responsivenss of the "job" to which they are applied by improving its own performance parameters (e.g., reducing its resource requirements). Thus the improvement is *independent* of the characteristics of other types of jobs. The synergistic principles, on the other hand, improve the overall responsiveness through *cooperation*: they can reduce the average time waiting for resources if the competing jobs abide by the recommended principle. The locality design principle is both independent and synergistic, because it can improve a job's own responsiveness as well as benefit competing jobs. As mentioned, the instrumenting principle is for control; it has no direct effect on performance, but is vital to managing it.

Most of the principles require a tradeoff decision. The performance of the various alternatives may not be obvious, particularly for the synergistic principles, since it is affected by many interrelated factors. A previous paper provided quantitative improvement formulas for the independent principles [SMI86b]. Performance models similar to Figure A.1 can aid in the decision making for the synergistic principles [SMI80a, SMI80b]. Because of the interaction, though, the definition of some of the principles may seem ambiguous. In practice, the performance engineering models (similar to that in Figure A.1) resolve the ambiguities.

Figure A.1



THE SYSTEM PERFORMANCE CYCLE: A METHODOLOGY FOR ARCHITECTING PRODUCT PERFORMANCE

Doug McBride
Hewlett-Packard Company
Performance Technology Center

ABSTRACT

Rather than wait until a computer software product has been designed and implemented to determine what its performance capabilities are, it is suggested that performance criteria be specified during the product investigation and design phases in order to ensure that acceptable levels of performance are realized at product completion. The methodology used to ensure performance goals are met, although straightforward, is not necessarily intuitive. This paper outlines the process known as "The System Performance Cycle", a set of guidelines used at Hewlett-Packard for the purpose of directing product teams toward producing software that performs when functionally complete, as opposed to having to go back and "tune" the product after performance shortcomings have been detected during post-release usage.

INTRODUCTION

Throughout the first three decades of the computer industry's existence, users of computer systems typically measured the efficacy of their machines by reliability (translated as "up time "), throughput (amount of "useful work" performed per unit time), and relative functionality (how "well" the system did the required work). These were the days of predominately batch processing where the end user had little or no contact with the system other than the submission of a work request and the receipt of the corresponding output within a certain period of time (usually the next day or beyond). The metrics used to define system performance were mainly batch, and the time frame was one of hours, or in the extreme,days.

Over the years advances in semiconductor technology made possible smaller, less expensive, and faster computer systems. These systems inspired a move toward "online" or "transaction" based processing where the end user interacted directly with the machine through some form of dialogue at a keyboard/display device. With this form of man machine interface came a new perspective on system performance; the emphasis became one of "how quickly" rather than "how much" (of course we know there is a relationship here, but the priority was on waiting the minimum amount of time for service by the system).

During the evolution of the online system concept, many installations experienced performance problems (poor response times) as the demand for system services grew and the complexity of the applications run on the machine increased. Initially, the solution to these performance problems was monolithic; buy a larger (read "more powerful") machine of the same family from your computer vendor and system performance improved commensurately. This worked early on due to most organization's adherence to the centralized data processing center concept. As technology made distributed processing via the smaller "work group" machine possible however, the performance-to-cost ratio could not be improved to the point of solving performance problems economically with hardware, especially in a machine family with a broad range of hardware capabilities.

At Hewlett Packard (HP), this concept became clear as the HP3000 family of computer systems grew in two directions; initially towards the "supermini" class of machine used to replace smaller "mainframe" type machines in transaction processing environments, and later the family grew into the smaller (10 - 30 user) "work group" class of machine to support OA and small business applications. Many of the software applications sold by HP and/or third party vendors ran fine on the large machine, but consumed inordinate resources on the small machines causing performance degradation severe enough to sometimes cause customers to look elsewhere for solutions.

A methodology needed to be put in place which would ensure software products would meet certain performance criteria prior to being released to the HP customer base. The methodology was created and refined by a group at HP Laboratories during the early days of research into the Hewlett Packard Precision Architecture to be used in the development of software systems for the new family of HP computer systems. These techniques have also been applied to the development of software for current systems as well as reworking existing software where performance is a problem.

Collectively, this set of procedures is referred to as the System Performance Cycle (SPC). The SPC has proven effective in architecting performance within the design of a software product primarily due to the inclusion of performance factors into all phases of the software life cycle from product specification through maintenance procedures after release. The remainder of this paper is intended to give an overview of the principal parts of the SPC as it is used at HP, as well as how it can be used in any organization to provide similar results.

ESTABLISHING GLOBAL PERFORMANCE GOALS

In studying how software was designed at HP, it was noted that performance was not an explicit line item at any level in the software lifecycle methodology. Functionality is stressed along with a high level of quality with emphasis placed upon what the product will do and how it will react under all circumstances. It is always "implied" that when the product is actually implemented, those who actually write the software will use all the resources at their disposal to ensure system resources are used efficiently and their subsystems will

perform "adequately" when actually used.

Actual experiences have shown that products designed without taking into consideration performance factors, or the interaction of the product with other products running on the same system, typically produce systems with unknown performance characteristics. Unless available system resources are plentiful or the demand placed upon the system is minimal, the interaction of the new product with others alter the overall characteristics of the system in question to the extent that predictable system performance is almost impossible.

In almost all cases, the acknowledgment of this phenomenon comes after product testing/implementation has begun. If the problem is severe enough, it becomes necessary to step back and examine what can be done to "tune" the product to maximize performance and minimize inefficient interaction with other subsystems. If testing techniques don't expose the performance problem during the QA phase, it is possible that the product could be released to customers with the performance quandary showing up at the user's site. In any case, this additional tuning activity is rarely accounted for in project schedules, causing schedule slippage and additional expense which reduce the overall effectiveness of the product to the company. In more extreme cases, poor product performance or severe system degradation can make the product a "non-product" in the user's eyes, causing a major redesign and rewrite in order to make the product viable in the selected marketplace.

The obvious solution to this problem is to include in the product design and specification phase a statement of the overall performance goals the product is expected to meet, along with a statement of system impact to facilitate the prediction of product resource demands on system-wide performance. The target customer (user) of the product should be solicited heavily during this stage to contribute to the overall (global) performance goals of the product, as the product would probably not be created if it were not for the needs of the users (let's face it - its their perception of the usefulness of the product which will determine the overall level of product success). From the description of required performance should come two very specific pieces of information which will be used repeatedly during the design and implementation of the product.

The first item of information (as previously mentioned) is a statement of overall product performance, stated in measurable, quantifiable terms. The customer may state their performance requirements in a somewhat inexact fashion, due to their unfamiliarity with data processing systems and terms. However, it is the responsibility of the design team to restate the performance goals in terms which will be useful parameters as the design phase progresses. This usually involves selecting metrics which are well defined, unambiguous, and above all, measurable.

Typically at HP we use Response Time and Throughput for these measures of product performance. The particular characteristics of the product under development will dictate the context in which these metrics are applied, as well as the definition of the transaction element which will determine how throughput is measured and evaluated. The definition of these terms may vary somewhat from installation to installation, but the importance is not in the precise definition (although the term should be defined for a particular instance), but in the consistency of application of the definition by all involved in the design and implementation of the product. The statement of overall product performance must also be qualified as to the environment under which the product will be tested for compliance to the goals.

The second item of information to be garnered from the global performance statement is a general measure of system resource consumption. Since performance is directly related to system resource availability, the behavior of the entire system environment can be estimated by:

- Understanding and quantifying the total amount of system resources.
- Understanding and quantifying the amount of those resources currently being consumed by existing applications and subsystems.
- Quantifying the additional demand which will be placed upon the system with the implementation of the new product currently under consideration.

This information is key to determining if the system under consideration for implementation of the application is suitable, or if additional resources are needed to successfully run the new product and cause a minimal impact on the operation of the other products and the overall perceived system performance parameters.

Given this data, a key decision point is reached prior to starting the implementation of the product; Can the product's global performance goals be met and still maintain acceptable overall system response? This decision can now be made long before a significant investment of resources has taken place, with commensurate savings if the decision must be made to alter the product strategy including the availability of certain system resources in order to meet product performance goals.

The points worth noting about this portion of the SPC can be summarized:

- It establishes and formalizes the user's expectations of the product's performance prior to detail product design and implementation.
- It causes the design team to take a much closer look at factors which will affect not only the performance of the new product, but the overall effect the product will have on the target system when implemented.
- By focusing on performance before the product is implemented, the resources necessary to support the performance effort are known beforehand, allowing better control of project resources and avoiding the situation whereby significant rework-

ing of the product becomes necessary due to lack of performance and/or excessive system resource consumption.

PRODUCT DECOMPOSITION

The next major step in architecting product performance comes during detail product design. The term "Product Decomposition" is derived from the exercise of determining functional modules within the product which can be measured more easily than the entire product, and will provide insight into where within the product potential performance bottlenecks may be occurring. Additionally, much like in structured design and programming, these modules are basically self-contained from a functionality standpoint and interface to the rest of the product through well-defined and documented input and output paths.

There are several reasons for the decomposition of the product to aid in performance analysis:

- The overall performance of the product can be stated in terms of its components, thereby allowing a more detailed examination of the factors that are affecting the observed product performance during development and testing.

- If a performance problem is detected in a specific module, the effort necessary to isolate it in a smaller module is less than in a larger unit of code.

- Individual modules can be assigned performance goals much like the overall product, allowing for easier modeling and tracking as design tradeoffs become necessary and as the product matures.

- If the product is modeled and formally tracked for compliance to predefined performance goals, the decomposition usually is performed at the level of granularity used for those functions.

Component-level performance goals are initially based upon the design team's estimates of how long the "path length" will be within a specific module (basically, how long the module will typically take to execute), and the time spent in any hardware or software locks and/or latches situated in the execution path. The preliminary figures are based upon either experience, measuring existing code which is similar in function to that which is to be implemented, or simple test routines coded and measured in order to get preliminary numbers to assign to a component.

A simple "paper model" is then created (although a spreadsheet is much faster) which predicts the overall product performance based upon the aggregate observed (estimated) performance of each of the components and their overall level of interaction. Depending upon the initial results of this predictive model, certain modules may require a revision of their individual estimates (based upon further investigation into the reality of achieving the revised performance goal). This process usually alerts the designers to areas of "sensitivity" which may dictate further decomposition or strict observation during the implementation phase of the product (especially if the initial indicators from the paper model show a large disparity from the global goals defined for the product).

The decomposition process becomes then an exercise which provides two major benefits in the design cycle:

- It identifies functional units within the overall product which provide key indicators of overall product performance and whose aggregate performance approximates the expected performance of the product, hence making them likely candidates for instrumentation and measurement.

- By explicitly stating the expected performance goals of each component in terms of "path length" (time in module) as well as time in locks and/or latches, summing the times and verifying a "close" approximation of overall product performance, an important check point is established prior to detail design and implementation which can focus additional resources on investigating potential performance bottlenecks.

MODELING THE PRODUCT

Once the product has been adequately decomposed into a group of functional components whose behavior on an individual basis is clearly understood, a more detailed model can be created to aid in design tradeoff and predictive analysis during product design and implementation. But - before going any farther on the subject - a gentle discussion on the "mystique" of modeling.

Just the mention of the term "modeling" turns a lot of people off. Modeling typically conjures up pictures of a PhD in Mathematics in front of a large chalk board busily writing with one hand and erasing with the other, all material being totally unintelligible to the average mortal. This is just about as true a picture as the old stereotype of data processing professionals wearing long white lab coats, carrying clipboards and wearing black, hornrim glasses. This is not to say it can't be (and sometimes is) that way, but the typical modeling effort does not have to be an exercise in post-graduate math or astrology (although the latter may help in some cases).

As mentioned in the previous section, a simple "paper model" or spreadsheet could be used to determine the aggregate product performance from the components the product was broken into for design and analysis. This is indeed a mathematical model, as it requires the relationship between product submodules be expressed in order to express estimated aggregate product performance. Typically however, even in the moderately complex simulation systems used to discretely model computer systems, all the math that is required is a working knowledge of queueing theory and statistics, as opposed to the underlying mathematics necessary to prove correctness of theorems.

Now, back to the subject of modeling the product's performance characteristics. How much farther a development team decides to go with a modeling exercise is mainly a function of product complexity versus accuracy of predicted results. Typically, we find that in order to achieve a consistently high degree

of accuracy in a model, the model must be more representative of the details of the actual system, and therefore may be inherently more complex. The more complex the model, the longer it takes to implement, and the higher the skill level required to implement, validate and maintain the model.

The time invested in creating the model must be offset by the benefits it will provide to the development effort. The strengths of models lie in the fact they are usually more easily implemented than the actual system under development, and can be used to answer questions key to design issues that affect product performance. In very complex products, a model is almost necessary to facilitate the understanding of module interaction and effect on overall system performance. In such a case, it is easy to justify the staffing and resources necessary to design, create, and support the product model. In simpler cases however, the decision to implement a product model is not so clear cut, and assigning important project resources to the effort may not be justified.

In all cases, however, modeling must be considered as part of the development effort which will have an effect on the outcome of how the product may ultimately perform. At a minimum, the simple "paper model" or spreadsheet must be used to sum component performance into overall product performance.

As the product becomes more complex, making design decisions which affect performance become more difficult without data to validate the decision. Models allow for experimenting with design alternatives before they are actually implemented, giving a fairly accurate idea of how the actual system will perform under similar circumstances. When given a choice of implementing actual portions of a product with the potential for redesign and recode due to performance problems, versus modeling the same circumstance ahead of the actual implementation, the choice becomes easier to make. A checkpoint should be established as part of the design effort which answers the model/no model question in all cases prior to actual product implementation.

TRACKING PRODUCT PERFORMANCE

As previously discussed, the steps of establishing global performance goals, decomposing the product into more manageable components and establishing component goals, and developing a product model are all measures intended to aid in the architecting of performance into the product, rather than measuring the product after implementation to see how well it performs. In order to allow managerial control over the development of the project from a performance standpoint, a mechanism needs to be put in place to allow for the tracking (monitoring) of progress towards meeting the established performance goals as the product is built.

A formal document is typically produced and maintained to monitor the progress being made towards meeting the formally established performance goals. This document is referred to as the Performance Tracking Document (PTD). The format of the PTD can vary from product to product, but it typically contains the following minimum information:

- A statement of the global performance goals established for the product. The goals are stated in terms of quantifiable, measurable metrics such as response time and throughput. These are the goals which are established at product specification time with interaction from the user/customer of the product.

- Definition of the environment under which the performance tests will be performed. This includes specific hardware/operating system configurations (and variations, if applicable), as well as the types of user/transaction mixes that will define how the product is exercised.

- A description of the components into which the product has been decomposed. Typically, a product pictorial model is included which shows the interrelationships and data/control flow among the various components. Additionally, a description of the functionality and performance expectations of each component is included.

- A detailed description of the test program/benchmark which will be used to drive the product during performance testing, along with any hardware/software configuration data for specific instances of the test (varying memory sizes, number of terminals, number of discs, etc.).

- A description of the measurement tools and component instrumentation used to determine how the product is behaving, along with methods for enabling measurements if they are not enabled at all times.

- A section whose primary purpose is to display projected performance versus actual performance over time. This is the true tracking mechanism, which at a glance should show progress on the overall product goals as well as the individual components. An area for remarks and comments should accompany the display of each component's performance history to explain why sudden shifts in performance (plus or minus) may have taken place.

- Usually an appendix is included to the PTD which describes the overall purpose of the product, the environment in which it will be used, and the importance of meeting certain performance goals in specific circumstances. This appendix continues to remind project management and performance tracking personnel of the "big picture" so the trees still stand out from the background of the forest.

The PTD is usually published at regular intervals from 2 weeks to a month, depending upon the frequency of updates required by management. Overall, the PTD provides a history of how the performance effort progresses (or in some cases, regresses) over the life of the product development cycle. It becomes quite obvious, for instance, when a major design change is implemented which affects performance either positively or negatively, as the change in component and product performance will be reflected in a time frame (via the PTD) which coincides with the implementation of the change. The PTD is also a forum where issues which will have an effect on product performance can be aired for

management perusal and can stimulate action to solve problems before they occur.

OTHER TOPICS

There are several other components which make up the SPC in addition to those discussed, some of more consequence than others. Three topics not covered in this paper which are part of the SPC and to which significant work has been devoted at HP are Performance Measurement Tools, Benchmarking Technology, and Workload Extrapolation and Characterization. Separate papers have been submitted which cover our efforts in the Tools and Workload areas and work is progressing on exposing our Benchmarking work in future efforts.

Figure 1 is a synopsis of the SPC showing the major points to which individuals or groups have been assigned to carry out the indicated tasks and the overall flow of the SPC as implemented at HP.

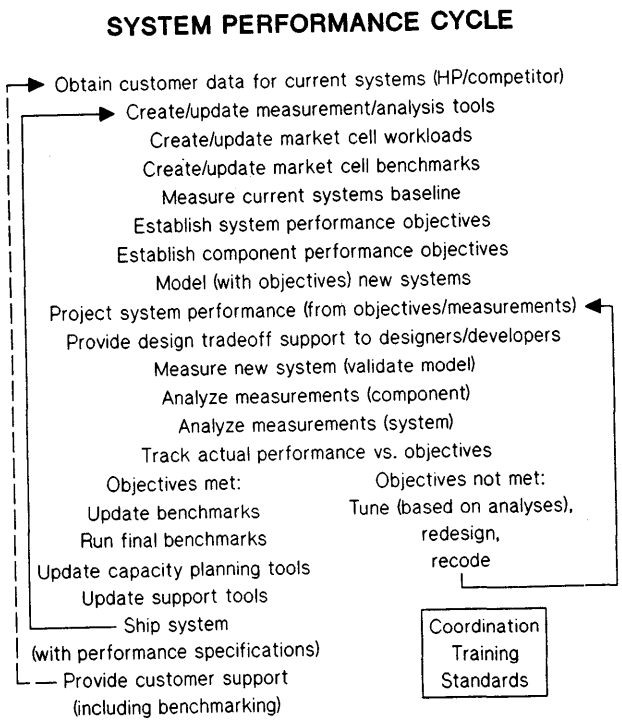


Figure 1.

FINDING PERFORMANCE PROBLEMS IN EXISTING PRODUCTS

Until all new products are engineered with performance in mind, we will likely continue to find areas within these products where performance problems are occurring. Besides, it may very well be that when the product was designed and implemented, performance was not an issue and the product performed in a satisfactory manner as far as the end user was concerned.

Well, times change, don't they? Additional terminals have been added to the system, the data base has grown and additional data sets have been added. The MIS department has decided to run background batch jobs at a low priority during the day because they feel it won't impact system response "that much", and a new set of applications have been added for the finance department. The performance of your product which at one time was considered adequate, is now driving the users crazy pressure is increasing to do something about it. What now?

Many of the principles applied to a new product can be applied to products which already exist. Global performance goals need to be established which the product must meet to be again acceptable to the users. Product resource consumption should be determined (an easier chore now...it exists to be measured!) and the resource demands of this product must be compared to the typical system aggregate resource demand to determine what percentage of the various system resources the product requires.

One of two possible paths can be selected at this point:

- Resource demand of the product appears to be in line with the work that is being done. This would imply that no significant gains in performance would be seen in relation to the effort expended to "tune" the product. Additionally, this usually implies the hardware/software system is "tapped out" in the current environment and additional analysis should be performed to alter the usage characteristics of the machine, or consider upgrading to a system with more resources.

- Resource demand of the product appears to be out of line with the work being done. Again, analysis can be performed regarding the usage patterns of the system, or a decision can be made to tune the product for better performance.

If the decision is made to attempt to tune the product, it should be analyzed and determinations made as to how to decompose it into smaller, more manageable components, just as the "new" product would be decomposed in order to aid in performance monitoring and design tradeoff analysis. The components should be measured, instrumented if necessary, and analyzed as to what particular part(s) of the product seem(s) to be contributing to the performance bottleneck and appropriate action taken to correct the problem.

Depending upon the level of undertaking, simple models can be developed to aid in major design change decisions to weigh the effect of the change on product performance, and to estimate the amount of system resources that may be required of released, depending on what is changed. Finally, if the tuning effort is major enough, a tracking document can be set up to allow the detailed monitoring of the changes made versus requirements to be met as the effort proceeds.

IN CONCLUSION

The thesis presented in this paper is a simple one; that a product can be developed from scratch with performance criteria specified beforehand, with a high probability that the performance goals established in the beginning will be met or the product should not be built.

From a practical standpoint, this concept makes a lot of sense. There just aren't too many products, no matter how great the functionality, whose "poor" performance can be tolerated over an extended period of time. It just costs too much money and has a tendency to develop emotional overtones after a while.

Therefore, why expend scores of man months pursuing the design and implementation of a product that will perform poorly and probably consume excessive system resources, if there is a good chance of the product being a failure? The effort is usually expended at some point to interject a certain level of performance into a product, although it is much more effective to do it at the beginning and carry it on through to completion, rather than to have the remark made during the preliminary acceptance test that "The product does everything I want it to, but I haven't got the time to wait for it to get done. Show it to me again when it's faster...".

"EXPERT SYSTEMS: LET'S BUILD ONE TOGETHER"

Frank J. Ingrassia
Innovative Management Solutions, Inc.

ABSTRACT

Talk is cheap, and there has been plenty of talk in the past few years about Expert Systems and their applicability to the Computer Performance Evaluation (CPE) area. I'll include myself among the great procrastinators, preaching AI, but doing much too little about it. The problem with Expert Systems can be stated in one sentence: they are very hard to build. Period. Knowledge Engineering is an art form in itself, and the languages of AI are difficult, if not bizarre, in which to write programs, especially for those used to Fortran, SAS, Assembler, et al. So, this paper is for everyone who wants to build an Expert System, but can't get past the first stumbling blocks. We will talk about organizing the knowledge, and the logic of the program, as we attempt to construct an Expert System for a CPE application. We will also use and discuss one of the AI languages, Prolog, and hopefully make some sense of it. It is my goal to unravel and simplify the construction of Expert Systems to the point where everyone will feel comfortable in building one. To accomplish this goal, I shall have to overcome the same obstacles we all stumble over. So join me now, as we tackle Expert Systems: let's build one together ...

INTRODUCTION: SIGNS OF LIFE NOTED

For more than two decades, academic and research institutions have explored the topical area of Artificial Intelligence (AI), from rudimentary (but not elementary) efforts to teach a computer to play a better game of chess, to more complex applications like medical diagnosis [1], geological analysis [2], self-psycho-analysis [9], and other "Expert" Systems. The author has previously explored the question of what constitutes an expert system, and the philosophical question of computer intelligence, in a prior paper [3]. Rather than continue to ponder what is ultimately in the eye of the subjective beholder, we will focus our attention on the construction of a real, live (?) Expert System.

The field of Computer Performance Evaluation (CPE) would seem to be a suitable domain in which to establish an expert system. Much is known about the cause and effect of tuning parameters, and there exist a plethora of rules of thumb (ROTs) and practical experiences from which to establish an impressive knowledge base for an "expert" on MVS, for example. Unfortunately, the data required to process even a subset of the MVS universe is quite substantial, as IBM researchers discovered. It turned out that more effort was required to develop, code, and maintain data reduction tools than was necessary to implement the inference mechanism and knowledge base in a recent IBM effort concerning the DASD I/O portion of MVS [4].

Since we wish to concentrate our efforts on building an expert system, rather than a data collection / reduction tool, we will choose an area of CPE that does not have data to input in machine-readable form. There are many fine data collection, reduction and analysis tools already established by prominent vendors in the CPE community. We shall leave the task of extending their work into the Expert Systems arena to these vendors, and take our departure into a more pristine environment: software engineering. The author has accumulated much experience in new application design and performance engineering concepts for large online systems, working with several modelling tools in a concerted effort to predict the performance and capacity of applications under development [5], as well as to illustrate the positive and negative impacts of engineering certain design changes [6]. For this study, we will select the universe of software engineering for applications under development, and concentrate on one domain within that universe, namely IMS DB/DC [7]. As an implementation vehicle, we have chosen the PROLOG language, an acronym for Programming in Logic. This choice was made predominately on usability and the widespread acceptance of this AI language in general. PROLOG is implemented for the mainframe as part of IBM's Expert System Development tools [8] under VM. Several versions of microcomputer compatible PROLOG are also available. The particular version we will be working with is TURBO PROLOG [10], a program product of Borland International. This version seems to be a very productive implementation, and comes with a variety of support tools called the TURBO PROLOG TOOLBOX [11]. Portability to the mainframe should hopefully be a minor task, especially in anticipation of IBM's obligatory announcement of PROLOG availability under MVS. Meanwhile, we can devote hours of R&D to our standalone IBM PC/AT, COMPAQ, or Compatible, rather than require extensive mainframe time, or the use of a special purpose LISP machine, to pursue our goal of building an Expert System. The ubiquitous presence of Personal Computers in all segments of our industry also helps to make this implementation strategy a practical choice. Besides, we are trying to unravel the mysteries of constructing expert systems: what better device than the common, harmless, garden variety PC ?

To begin with, we must approach this expert system construction with some specific goals in mind. We are concerned about the design of IMS online systems, but we haven't specifically said what we want to accomplish in that domain. Our goals are these:

1) to construct an expert system knowledgeable in IMS internals and current hardware;
2) build a user-friendly interface to solicit IMS transaction design information;
3) calculate the expected response times, throughputs and utilizations;
4) evaluate the results vs. user-defined performance objectives, offering recommendations and trade-offs on design issues, to achieve those objectives.

An ambitious project, to say the least ! Let's also say what we are NOT trying to do: we are not trying to develop another modelling tool. Rudimentary (read "simple") performance calculations are all that are planned. We are much more concerned with making intelligent design choices, not by trial and error, not user-driven, but DATA DRIVEN. If all works according to plan, the results of design input will dictate what recommendations and design alternatives are presented. Realistically, we do not expect to complete all the work in 1987. However, enough has or will be achieved to demonstrate the viability of expert systems technology in solving real problems in Software Engineering and CPE. Let's now begin: we'll learn together.

PROLOG, THE LANGUAGE: PROGRAMMING IN PURGATORY

Forget what you know about conventional programming: none of that will help you master the PROLOG language. In fact, the author is convinced that those fresh out of

college, or non-DP altogether, have a distinct advantage in learning to code in PROLOG! The language itself is devoid of the things we take for granted in COBOL, Fortran or PL/1. Simple things, like GOTO statements, or DO loops, for instance: you will not find them in PROLOG. There is likewise no assignment statement! The simple counter X=X+1 is not available. PROLOG is different: it is not a procedural language at all. We don't give PROLOG instructions: "do this, do that, then do this". Rather, we give the program "facts", and operational "rules" concerning those facts, and it figures out how to solve the problem.

The TURBO PROLOG compiler is a "backward chaining" implementation. In this methodology, IF-THEN rules start out with a goal and proceed backwards until all the premises satisfying the goal have been proven. This is sometimes referred to as "goal driven reasoning", because the process is driven by the goal it attempts to prove. Other AI systems use "forward chaining", where the IF-THEN rules are applied to facts in forward motion, until the facts culminate in a recommendation. That method is also referred to as "data driven reasoning", because the solution is achieved by the existence of an initial set of data. There are a few other concepts to master as well. As stated, there is no formal assignment statement in PROLOG. Instead, a pattern matching process called "unification" gives values to its variables. The variables can be considered either "free" (having no value) or "bound" (bound to a particular value), and often oscillate between these two states as PROLOG moves from clause to clause. The "clause" is PROLOG's equivalent of a statement. It is written in sentence form, usually presenting an IF-THEN case, only backwards. The conclusion comes first, followed by the conditions. For example,

```
domore if write ("Another?") and readln (Ans)
          and Ans=y.
```

This says that "domore" will be true (succeed) IF the message "Another?" is written on the screen, AND a response is read (Ans), AND that response was a "y" for Yes. Isn't this fun? Obviously, "domore" is a condition in some other clause which will only succeed if all these conditions are met. This is what PROLOG is all about: truth (or falsehoods) and succeeding in your clauses, leading ultimately to your goal. The language has a few other idiosyncrasies worth mentioning. Symbols have no meaning. All PROLOG symbols and clauses and "predicates" (action verbs) are written in lower case. Only Variables are written in upper case. The language uses "recursion" to allow clauses to be both input and output driven. Remember our mention of "free" and "bound" variables. A PROLOG clause can be used to test if a condition exists (or is true) or to prove a goal because the relationship (or clause) exists. For example,

```
calc(x,y,z) if X=4 and Y=Z/X.
```

If this clause is referenced with Y free and Z bound to some value, then X will be 4 and it will succeed. However, if called with X bound to 3, it will surely fail because X must =4 as one of the conditions for truth. Thus, the clause above could perform the calculation OR test whether X has the value 4. Admittedly, a crude example, but it will make more sense, in light of the discovery that PROLOG allows multiple clauses of the same name and executes them, in sequence, trying successive clauses as the prior ones fail. Thus, we could have 5 successive "calc" clauses, with X having values of 1 through 5, if that was our intention.

When searching for clauses that meet the goal of the program, PROLOG will use a technique called "backtracking". This occurs when subgoals (conditions which are part of the goal) succeed, but others fail. PROLOG will backtrack to the last successful subgoal (or condition) and try another set of clauses. You can control backtracking via the cut symbol ("!"), telling PROLOG that you are satisified with 2 out of 3 conditions, for example. You can also force PROLOG to keep repeating a clause, using recursion, until it fails, by using the special predicate "fail". This might be helpful to print a list until there is nothing left to print, for example. The language composition includes five main sections:

1) DOMAIN — where descriptions of PROLOG objects occur, including their type (character, integer, real, string, etc.). This is equivalent to DECLARE in PL/1 or INTEGER * 4 in Fortran.

2) DATABASE — where those variables/predicates which will make up the knowledge base are stored. This is equivalent to COMMON in Fortran, an incore dynamic data storage area.

3) PREDICATES — the "form" of subsequent clauses must be declared, so that PROLOG can know the relationship of the data used in the clause. This is akin to declaring a FUNCTION in Fortran.

4) GOAL — this is the ultimate objective the program is trying accomplish, or prove. The goal may be complex, and include many subgoals which must also be satisfied.

5) CLAUSES — are sentence oriented statements of the language. Here, "Rules" and "Facts" are described, which are then used by the program to solve its goal.

The TURBO PROLOG implementation used also contains many standard "predicates", which perform useful functions. The "fail" predicate has already been noted. The following are some predicates that perform and control I/O:

```
makewindow    - draws a window on screen
clearwindow   - erases contents of window
removewindow  - removes a window from screen
cursor        - positions the cursor at x,y coordinates
window_attr   - changes color attributes for window
shiftwindow   - moves to another window
scr_attr      - sets a character position's attributes
scr_char      - reads/writes a character at position
field_str     - read/writes a string of characters
field_attr    - sets a string's attributes on screen
write         - displays data on screen or other device
writef        - formatted output writer
readln        - reads a line of data (string or symbol)
readint       - reads integers
readreal      - reads real numbers
readchar      - reads a single character
file_str      - reads the contents of an entire file
nl            - skip to a new line
```

TURBO PROLOG has a full complement of arithmetic operators, and several useful Built-in functions, including:

ARITHMETIC OPERATORS: +, -, *, /, (,), <>, ><, $hex

RELATIONAL OPERATORS: >, <, >=, <=, =, <>, ><

PREDICATES: bitand, bitor, bitnot, bitxor, bitleft, bitright

BUILT-IN FUNCTIONS: mod, div, abs, cos, sin, tan, round arctan, ln, log, sqrt, random, exp.

TURBO PROLOG has a fair complement of graphics capability, including:

```
graphics      - set mode CGA/EGA
dot           - draw points
line          - draw lines
```

```
penup          - lift the pen (turtle graphics) *
pendown        - place the pen
pencolor       - color the pen
forward        - move it forward
back           - move it backward
right          - move it to right
left           - move it to left
```

* NOTE: turtle graphics simulate "joystick" control of an imaginary pen on the screen.

TURBO PROLOG has file I/O capability as well, in the following predicates, which also operate on any other device (PRN,COM,etc):

```
openread       - open a file, for Read only
openwrite      - open a file, for Write only
openappend     - open a file and append to the end of it
openmodify     - open a file for read/write
closefile      - close a file
readdevice     - read from file
writedevice    - write to file
filepos        - position pointer in file at record,byte
existfile      - test for file's existence
deletefile     - erase a file
renamefile     - change file name
eof            - test for end of file condition
flush          - flush the file Buffers
```

TURBO PROLOG offers many ways to manipulate strings and lists; some of these predicates are:

```
frontstr       - split a string, peel off front "n" chars
str_len        - finds length of string
isname         - tests if string is a valid PROLOG name
frontchar      - pick off front character of string
fronttoken     - split string into a list of tokens
char_int       - convert character to integer
str_char       - convert string to a single character
str_int        - convert string to an integer
str_real       - convert string to a real number
upper_lower    - convert upper to lower case letters
```

There are several miscellaneous predicates available, which do some useful things for the programmer:

```
findall  - find all occurrences of a variable in a List
free     - frees a variable, or tests if free
bound    - tests if a variable is bound
inkey    - reads a single char from any input device
beep     - produces the TTY "Bell" equivalent
sound    - produces a range of sounds
dir      - will produce a DOS file directory
concat   - concatenates strings
time     - returns current time of day
date     - returns current date
```

Finally, there are the database (knowledge base) controlling predicates, of the utmost importance in manipulating goals, clauses and searching for solutions:

```
asserta  - inserts a fact before all others in dynamic
           database
assertz  - inserts a fact after  all others in dynamic
           database
retract  - removes a fact from the database
save     - saves  the  dynamic database into a file
consult  - reads a file and restores the dynamic data-
           base
readterm - reads facts from a file
```

There are several syntactical things to know. TURBO PROLOG uses some special characters:

```
:-   if
,    and
;    or
!    cut
|    splits head/tail of a list
[]   a list
*    list follows
_    anonymous variable
```

and the word not, implying negation. A full explanation of the syntax and uses of TURBO PROLOG language may be found in the user's guide [10]. Noteworthy is the entire framework around Borland's TURBO PROLOG facility. It contains its own EDITOR, and TRACE facility, essential for following the logical path of your programs. The compiler also generates standalone, EXECUTABLE object code, in contrast to most interpretive versions of the language. A highly recommended companion product is the TURBO PROLOG TOOLBOX [11], which contains a number of canned PROLOG programs which may be "included" into your own program to process windows, menus, status lines, context sensitive help screens, DOS and directory operations, BIOS calls, and other useful functions. A veritable cavalcade of sample programs and usable source code come with each of the products, which will get users on their way in no time. Before closing this section on Prolog, The Language, let me point out that Borland's TURBO PROLOG is but one of several microcomputer versions of the language available. As a starting point for those interested in acquiring a PROLOG compiler, there exists a fairly recent comparison of 14 different PROLOG compilers conducted by AI Expert [15], which is worth reviewing. As for TURBO PROLOG specifically, there are some pitfalls and limitations that you should be aware of:

1) Don't use too many variables in a single clause: 100 is the maximum. You can get around this by calling another clause from the primary clause.

2) Close windows when they are not in use, especially Toolbox #80-85, which can be created multiple times: about 22 open windows is the maximum. A simple set of clauses like the ones below may be helpful, which close all windows except window #1, the primary screen:

```
closeall:- shiftwindow(X), X<>1, removewindow, closeall.
closeall:- shiftwindow(X), X=1.
```

3) The Trace compiler directive adds about 200 16-byte paragraphs to memory requirements, which may need adjustment via the CODE=nnnn compiler directive.

4) Use the Dynamic Database incore to pass data around, rather than having clauses with many parameters. This works like a COMMON area in Fortran.

5) Data may be passed to clauses called by other clauses. This is like Subroutine calling in other languages, as data values may be passed to and returned from the clauses invoked.

6) Use Menu processing to provide external control of the program's execution flow. Menu selections may be hard-wired to a trigger clause, which in turn invokes a series of clauses: the menu is the goal, and can always be satisfied; the menu options are the sub-goals, which you may choose to solve.

7) Use CUT (!) and FAIL judiciously, to prevent memory or stack overflow conditions. CUT will stop processing as soon as enough of the clause's conditions succeed. FAIL will invoke recursion until the clause fails, then move on.

8) Break up large programs into smaller, more manageable ones. PROLOG has memory limits equivalent to RAM size. Use the dynamic database to pass data & facts completed so far to the next program in a chain, via save

and consult.

9) Use TRACE to follow your logic; use the DIAGNOSTICS compiler directive to see how much memory you are using, as well as which variables / predicates are not being actively used in the code. Use the "storage" predicate to dump the current Stack, Heap and Trail sizes: manage memory efficiently.

10) Play and learn with one program at a time. There are literally hundreds of sample programs provided, and each demonstrates a particular feature or routine. For example, how to write lists of "n" variables, how to build a database manager, how to append things to a list and find them later, how to build accumulators, screen generators and controllers, how to do file I/O, etc. Master one at a time and don't write numerous of lines of code before you start testing: keep it simple and you will be most productive, and test each new clause of the program as it is added, using TRACE facilities to ensure that it does what you require. See examples of source code provided in the next section for further illustrations of these techniques.

KNOWLEDGE ENGINEERING: KNOWLEDGE IS POWER

Having completed our excursion into the semantics and facilities of PROLOG, we now turn our attention to the task at hand. Our first major task is at the data level: we must design our Knowledge Base for IMS DB/DC. In all seriousness, this is a trial and error task requiring considerable time and effort. We must go through successive iterations of the database design as we begin writing and testing the code. The format of the data and data types have significance. Shall we use numbers indicating different scenarios, i.e. 1=MPR, 2=BMP, 3=WFI, 4=EMH, or shall we use symbols (or character strings) to represent the data? The final choice, it turns out, is largely a matter of programming style. If you employ menu selection, for example, PROLOG returns the choice number you selected on input. Now, you can convert this number into its equivalent symbolic representation, or just look up the number! Designing an efficient knowledge Base is very similar to the task a Database Administrator (DBA) might perform for IMS. Questions like "Is this data element absolutely necessary?", or "Can this element be calculated from other elements?", and similar situations need to be addressed. Considering that all of the knowledge base may need to be memory resident at some point in time, or stored in files, we should be conscious of the same storage efficiencies as with MVS. Data is also grouped into fact clauses by combining similar and related elements, just like in other databases. Another decision must be made regarding the data location itself: should it be hard-coded into the program, as clauses, or stored in a file for consultation later? Similarly, knowledge acquired by executing the program (read "created at runtime") also needs to be thought out, and designed, just as the permanent knowledge base. There is no magic secret to designing an efficient knowledge base, but it is best accomplished by observing three rules:

1) keep related data in the same clause;
2) eliminate redundancy and calculatable elements;
3) shorten data formats wherever possible, using symbols.

Our IMS expert system employs several databases, and three main program components. The first program, CMGDESC, is used to solicit the design of transactions from the user, who describes what the transactions will do. These descriptions of the transaction profile are created and stored in an internal database, which is saved into a file at the conclusion of the program. This file, a Knowledge Base describing the user's transactions, becomes the input Knowledge Base for the second program, CMGCALC, which calculates the path lengths and expected performance of the transactions. These are then added to the Knowledge Base, and passed (via file) to the third and final program, CMGANAL, which analyzes the calculated performance vs. the user's performance objectives, and makes decisions and recommendations for improving the design. These three programs are used in concert to satisfy our overall goal of designing efficient IMS transactions. See Figure 1 for the logical flow of the data and the Knowledge Databases employed. They are three (3) separate programs to economize memory.

Now that the data flow has been ascertained, and the various internal and external databases identified, let us look at some sample "knowledge" contained in each database. Examining Database 1 (from Figure 1), the Internal Hardware characteristics database, we find knowledge of the form:

```
                cpu_char (cpuname, #engines, miprate)
      e.g.,     cpu_char (3090-400E, 4, 61.50)
```

to describe various CPU characteristics, needed for our response time and capacity calculations. Also in this database are device characteristics, of the form:

```
                dev_char (devname, seek, transfer, revtime)
      e.g.,     dev_char (3380, 16, 3.0, 16.7)
```

Additionally, certain default numbers for IMS datasets and system libraries, like Buffer Hit Ratios, Syslib Blocking Factors and other similar information are contained herein, to form a default pool of IMS options in a well-tuned environment.

Database #2 is the PROJECT database created by CMGDESC which describes, in PROLOG terms, the Transaction Life Cycle. Many projects may be examined in our conceptual system design, so we'll give each a project name, and qualify all datasets with ".PRJ". Update capabilities are an obvious future consideration, based on the results of our analysis by the "expert". Database #2 is the epitomy of PROLOG structure in its purest form:

```
                project (projectname, status)
      e.g.,     project (CMGDEMO, 0)

                lso (type)
      e.g.,     lso (S)

                logging (option)
      e.g.,     logging (dual)

                thecpu (cpuname, #engines, miprate)
      e.g.,     thecpu (3090-400E, 4, 61.50)

                adevice (devname,seek,transfer,revtime)
      e.g.,     adevice (IMS001, 16, 3.0, 16.7)

                afile (dbdname, accmeth, blksize)
      e.g.,     afile (Myfile01, HIDAM, 4096)

                indexfor (ixname, dbdname)
      e.g.,     indexfor (Myindex1, Myfile01)

                liveson (dbdname, devname)
      e.g.,     liveson (Myfile01, IMS001)

                afpdb (dbdname, option)
      e.g.,     afpdb (MyMSDB01, Pagefix)

                runsin (txclass, region)
      e.g.,     runsin (4, Region01)

                runsat (region, priority)
      e.g.,     runsat (Region01, 16)

                argn (region, txclass 1/2/3/4)
      e.g.,     argn (Region01, 12, 10, 2, 1)
```

```
          atlc (tlcname, tps, pgmsize,  region,
                acbopt, lang, preload, inquiry,
                mode, imstype, txclass, PLtype,
                pathlen, insegm, insize, inflds,
                outsegm, outsize, outflds, spaloc,
                spagu, spais, tpaccess)
e.g.,     atlc (Tran001, 100, 512, MPR, DOPT,
                COBOL, YES, NO, SNGL, FF, 12,
                LOC, 12000, 2, 1024, 50, 4,
                2048, 100, CORE, 2, 2, VTAM)

          adbd (tlcname, dbdname, usesix, gu, gn, is,
                ir, ds, dr, de, re, op, pos, fld)
e.g.,     adbd (Tran001, Myfile01, YES, 2, 4, 2,
                 0, 0, 0, 1, 1, 0, 0, 0)
```

As you can see from the preceding examples, we have stated the description of our various transactions in sentence - like form. Most important to our work is the definition of "atlc" and "adbd", which define the transaction life cycle (tlc) and database descriptions (dbd) respectively. Those familiar with IMS will notice many familiar acroynms and options above: we will not explain them further here. Suffice to say that they are IMS-specific, and carefully selected.

Database #3 is an internal database representing the actual pathlengths for various IMS system activities, as one might find in CRYSTAL [12], SNAP/SHOT [13] or other [14] modelling tools. As this data is largely proprietary in nature, actual values cannot be included herein. However, the form of our third database is representable, as follows:

```
ims_msgin (tpaccess, inquiry, P/L)
ims_msgout (tpaccess, inquiry, P/L)
ims_sched (regiontype, P/L)
ims_acbload (regiontype, acbopt, P/L)
ims_pgmload (preload, language, P/L)
ims_appl (P/L Type, Language, P/L)
ims_mfs (in/out, P/L Flds, P/L Segm)
ims_spa (spaloc, function, P/L)
ims_iopcb (mode, function, P/L)
ims_dli (accmeth,function,six, SP/L, XP/L, YP/L, NP/L)
refindex (usesix, factor)
ims_fastpath (accmeth, function, P/L)
ims_log_phy (accmeth, function, ratio)
ims_buffpool (area, fix P/L, nofix P/L)
ims_fixbuff (area, ratio)
ims_pio (accmeth, fix P/L, nofix P/L)
ims_log (P/L)
ims_iofactor (area, ratio)
```

The above list is a nearly complete summary of the types and formats of the IMS path length internal database, also illustrating the relationships between the data elements.

Database #4 is the output of the CMGCALC performance calculation program. It contains summary level data which is the result of applying the path lengths to the transaction descriptions, and calculating simplex service times, response times and utilizations. Some of its records resemble the following:

```
          exec (tlcname, txperhr)
e.g.,     exec (Trans001, 8000)

          inclass (txclass, tlcname)
e.g.,     inclass (12, Trans001)

          uses (CPU, tlcname, svc)
e.g.,     uses (3090-400E, Trans001, .402)

          doesio (dbdname, tlcname, I/O count)
e.g.,     doesio (Myfile01, Trans001, 12.2)

          fileio (dbdname, %utilized)
e.g.,     fileio (Myfile01, .03)

          cpuse (tlcname, %utilzed)
e.g.,     cpuse (Trans001, .044)

          devuse (tlcname, devname, svc)
e.g.,     devuse (Trans001, IMS001, 10.382)

          devio (devname, %utilized)
e.g.,     devio (IMS001, .092)

          filuse (tlcname, dbdname, svc)
e.g.,     filuse (Trans001, Myfile01, .938)
```

The above are some samples from the 4th database, which summarize a transaction's impact on the system configuration proposed. CMGCALC takes in descriptions of the Transaction Life Cycles, and attendant DBD, REGION, IMS System and Configuration data, and calculates expected results in this format.

The final database, CMGANAL.DBA, will contain the IMS design analysis rules, to be applied to the summary AND detailed descriptor information collected and asserted by CMGDESC and CMGCALC. These tuning rules will analyze the design and match the calculated results against user - supplied PERFORMANCE OBJECTIVES. As of this writing, formulation of the 5th and final database is incomplete.

Anticipated rules will operate on several levels:

1) General Configuration Efficiency
   - CPU utilization
   - device utilizations & data rates
   - file utilizations

2) IMS System Efficiency
   - region/transaction distribution
   - region occupancy time
   - IMS Syslib utilizations
   - Concurrency and overheads

3) Application Program Efficiency
   - overhead functions for system services
   - detailed I/O analysis
   - file and access method effectiveness
   - transaction profiling and characterization

4) User Objectives & Expectations
   - throughput analysis
   - response time analysis
   - workload decomposition

By applying facts and rules in the above categories, it is anticipated that we should be able to conduct a relatively complete analysis of the proposed TLC design, and recommend some appropriate modifications in areas deemed to produce the greatest benefits in performance efficiency.

DEVELOPING THE EXPERT SYSTEM: ROME WASN'T BUILT IN A DAY

As previously stated, the purpose of our Expert System is to evaluate proposed designs of IMS transactions and to recommend improvements and enhancements based on user - defined objectives. This is our target system, still under construction. However, the purpose of this paper is to illustrate how such expert systems might be constructed. We have spent considerable time explaining the facilities available in TURBO PROLOG, as well as the design and content of our Knowledge Base. This section deals with program construction, and contains samples of working code which illustrate the implementation strategy of our proposed system.

We have made mention of using menus, both to gather data in a user friendly fashion, as well as to direct PROLOG to address major subgoals, or program sections, in our system. Figure 2 shows the entry menu, the primary goal (run) and how the various choices in the menu invoke

subgoal processing. Menu choices can be executed at multiple levels. Selecting option 1, "Create a Project", invokes the build predicate, which in turn offers a second menu as defined in Figure 3. This menu presents the user with a choice of various things to describe for the Project, including:

- Setup IMS Sys Gen
- Set Configuration
- Database DBD Gens
- Define Transaction
- Define Regions/Mix
- Save Project & Exit

Construction in this form allows the user to enter any number and any order of descriptions. He or she may skip the IMS gen, enter a few transactions, define a region, go back to transactions, etc., and save the project database whenever suitable. Other primary menu options will allow the updating of partially completed projects later. Users may even redefine certain values which have only one definition, and the latest definition will always take precedence by utilizing PROLOG's asserta database insertion device, at the front of the dynamic database.

The simple task of gathering user data about the design, and the inference of that data, is illustrated in Figure 4, the defrgn (define region) predicate. Notice the use of TURBO TOOLBOX tools (changestatus, lineinput_leave), as well as standard PROLOG data drivers (makewindow, write, readint). Most important is the asserta section, which stores the knowledge acquired into the incore database. Finally, observe the looping construction of morergn: it returns to defrgn execution if the answer to "Another RGN?" was yes; else, it fails and drops to the second morergn clause, which invokes the closeall windows routine.

In Figure 5, the parameter data, in the form of variables, is passed to the chkindex clauses. If USIX="y", meaning that secondary indexing is requested, then the routine proceeds to gather additional data about the index, and makes the appropriate assertions to the database. However, if this clause fails, the next chkindex clause is tried, and succeeds with USIX="n", or no secondary index. It then becomes basically a NOP instruction, taking no further action because none is necessary. Another useful feature of PROLOG is it's ability to work with "lists". Consider the use of findall in the setdevice clauses in Figure 6.
It searches the internal knowledge base for "dev_char" clauses, and compiles a list of their device names. The list is written by the wrlist predicate, also in Figure 6. It's purpose is write 7 devices at a time, skip to a new line, and write 7 more, until the list is empty. Notice we also provided the user with a capability to define new devices using the DEF command. In getdev clauses, if Adev = "DEF", a new device, and it's characteristics, are solicited from the user. Otherwise, the dev_char clauses are searched for a match against the device type, and asserted if found. Lastly, we fall through to a third getdev, with an "anonymous" variable (_, meaning any value) for Adev: if you come this far, you probably spelled something wrong, so back-track to disks and start over.

PROLOG also has arithmetic ability, as displayed in Figure 7. Here, code taken from CMGCALC shows a portion of the pathlength calculations for a transaction life cycle. It consults the "ims_log" portion of the knowledge base, asking PROLOG to reference the knowledge base and return the value of LREC, a path length for physical logging. It then performs some calculations on other variables, using LREC in the process, and completes with a call to append, a routine which adds things to a list (in this case, a calculated value LOGREC, appended to the back of LIST8, to form LIST9). Note that these are just some of the clauses in the dotlc predicate, which calculates the entire pathlength of a transaction, using the sumlist clauses.

These excerpts from working code written in PROLOG are included to illustrate the relationships of data and predicates, and the power of the language itself to express complex manuvers in simplistic terms. In all fairness, PROLOG is not an easy language to understand. However, with practice, a great deal can be accomplished in a few lines of code. The author estimates that 5 to 10 times fewer instructions are required by PROLOG than by other procedural languages to perform the same task. As part of our investigation into PROLOG's capabilities, a complete database management system, with full screen input, update search by key and report generation was built, tested and operational in less than 3 days: that's productivity! However ...

As hinted in the section heading, the thorough development and testing of an expert system is not an overnight project, regardless of the power and flexibility of the language. At this stage, CMGDESC, the IMS Transaction descriptor program, is complete, tested and operational. It can be used to describe any IMS transaction, including Fast Path, and constructs the proper data driven knowledge base. The second program, CMGCALC, is also working, and in the final stages of testing and verification. The final component, CMGANAL, is still in design mode, as is the database of tuning rules it shall employ. In retrospect, the hardest part of building expert systems can be summed up in one statement: Knowledge Engineering. Still, we have come a long way toward demonstrating the viability of expert systems in solving today's problems, and we are quite satisfied with our results to date. Given more time and effort, we intend to learn even more in the future, and look forward to the eminent transition of AI from research to practicality. As the saying goes: "Two out of three ain't bad !"

ESOTERIC ISSUES: I THINK, THEREFORE I AM ?!?

The author would be reticent not to express a few comments on the potential and future uses of expert systems technology. It is obvious at this point that existing systems can be rewritten and duplicated using AI, with minimal effort. However, that is not much of an accomplishment. We need to push the limits of imagination further, and contemplate new uses for this emerging opportunity. We know that the knowledge of several experts can be pooled, making the resultant AI system smarter than any one of it's human contributors. More work needs to be done on the pure heuristics of self-learning and adaptation. Self-learning methods to date have been in the form of historical success/failure tracking, teaching systems to learn from their mistakes or triumphs. Further techniques are required that can emulate the ability to be inquisitive perhaps, or evolve in understanding. At some point, expert systems must have the ability to request more data, leading the conversation, rather than being driven by our limited goals. Perhaps a freestanding, freethinking expert system, concentrating on a suitable knowledge domain, might someday reconstruct the patterns of it's facts and rules to arrive at new conclusions we had not anticipated. As cycles and parallelism in CPU's become more abundant, it may be that our greatest opportunity is to program an expert system with rules about how to acquire knowledge and combine concepts, and let it crank away, searching for new ideas and formulating new data relationships that are beyond our own capability to establish. Ultimately, emotional response, personality and introspection might be envisioned, with a dash of humor and a dose of morality, which might just culminate in a complete, free thinking, Actually Intelligent system. We stand at the threshold of discovery: let's make the most of this rare opportunity to be creative.

REFERENCES

[1] MYCIN, and EMYCIN, developed at Stanford University.

[2] PROSPECTOR, developed by SRI International.

[3] "Artificial Intelligence: Building the Perfect Beast," F.J. Ingrassia, Computer Measurement Group Proceedings, 1985.

[4] "Prolog for Applications Programming," W. G. Wilson, IBM Systems Journal, Vol. 25 No. 2, 1986, G321-0083.

[5] "The Role of Capacity Planning In Software Development", F.J. Ingrassia, Computer Measurement Group Proceedings, 1982; IBM GUIDE 56 Proceedings, 1983.

[6] "Modelling the Performance of New Online Systems," F.J. Ingrassia, Computer Measurement Group Proceedings, 1985; Journal of Capacity Management, Vol. 3 No. 1, 1985.

[7] Information Management System/Virtual Storage IMS/VS, IBM Program Product 5740-XX2, IBM Corporation.

[8] "Introduction to IBM's Knowledge Systems Products", A.J. Symonds, IBM Systems Journal, Vol. 25 No. 2, 1986, G321-0083.

[9] ELIZA, "Exploring Artificial Intelligence on your PC", T. Hartnell, Bantam, 1985.

[10] "Turbo PROLOG: Owner's Handbook", Borland International, 1986.

[11] "Turbo PROLOG TOOLBOX: User's Guide and Reference Manual", Borland International, 1987.

[12] CRYSTAL, program product of BGS Systems, Inc., Waltham, MA.

[13] SNAP/SHOT, internal marketing aid, IBM Corp., Raleigh, NC.

[14] ANIMSVS, internal marketing aid, IBM Corp., available through IBM's HONE system.

[15] "PROLOG For The People", L. Burkholder, et al, AI Expert, Vol. 2 No. 6, June 1987.

FIGURES

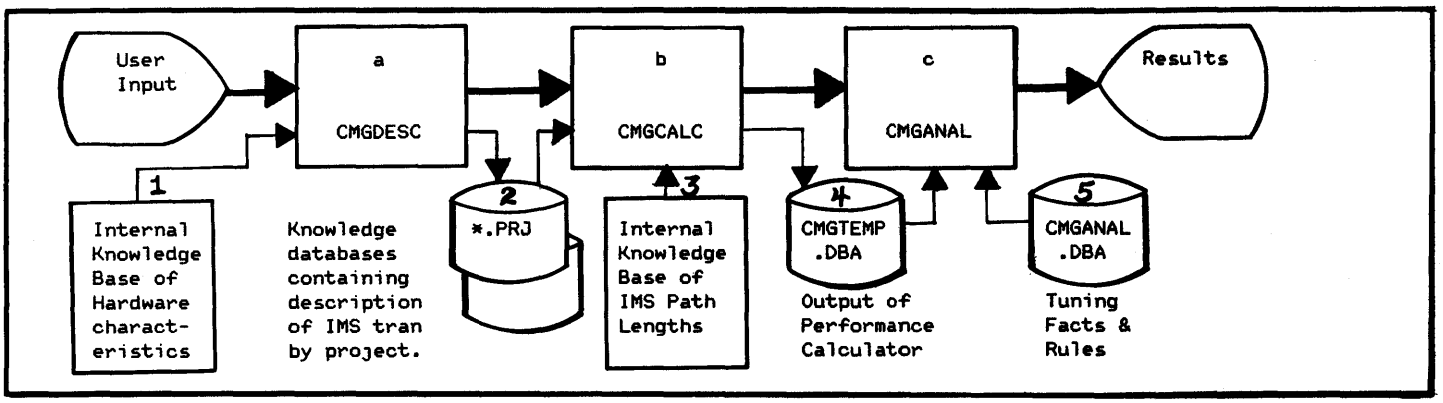


Figure 1. Overview of Proposed IMS Expert System

```
goal
    border(11),makewindow(1,31,28,"IMS Design Advisor",0,0,24,80),
    makestatus(95,"   Use Arrows to select the desired option"),
    run.

clauses
        run :-  repeat,boxmenu_leave(5,27,5,20,15,11,
                [" Create a Project"," Update a Project",
                 "Display a Project"," Delete a Project",
                 " Exit the Program"], "PROJECT MENU",1,PCHOICE),
                 project(PCHOICE), endproj(PCHOICE), !.
    project(0) :- exit.
    project(1) :- build.
    project(2) :- updproj.
    project(3) :- showprj.
    project(4) :- droproj.
    project(5) :- nl,write("\1\2\1 Thanks for using the IMS Advisor
                  \1\2\1 \n\n ...Press Any Key To Exit..."),
                  readchar(_), exit.
    endproj(0) :- clearwindow, write(" Exit Requested ! \n (press
                  any key) "), readchar(_), exit.
```

Figure 2. Entry Menu and Primary Goal

```
     build :- shifwindow(1), clearwindow,
              changestatus(" Create: Select part of project to modify"),
              boxmenu_leave(5,27,6,22,11,14,
             ["Setup IMS Sys Gen","Setup Configuration",
              "Database DBD Gens","Define Transactions",
              "Define Region/Mix","Save Project & Exit"],
              "CREATE MENU",1,CH11), part(CH11), build, !.
    part(0) :- removewindow, run.
    part(1) :- imsgen.
    part(2) :- configure.
    part(3) :- dbdgen.
    part(4) :- deftlc.
    part(5) :- defrgn.
    part(6) :- changestatus(" SAVE: Put Descriptions in a File "),
               lineinput(20,22,29,3,14,"Enter Project Name: ",
               "projxxxx",Project), frontstr(8,Project,PROJ,_),
               asserta(proj(PROJ,0)), concat(PROJ,".prj",Projfile),
               saveproj(Projfile), removewindow, !, run.
```

Figure 3. Secondary Menus: User Directed Flow

```
    defrgn :- shiftwindow(1), clearwindow,
              changestatus(" Define IMS Regions & Tran Classes"),
              lineinput_leave(5,1,30,31,31,"Enter Region Name: ",
              "region01",RGN), frontstr(8,RGN,Region,_),
              lineinput_leave(5,40,30,27,27,"Dispatching Priority:",
              "16",Dis), str_int(Dis,DPRTY),
              makewindow(15,30,30,"Tx Classes",12,30,7,20),
              changestatus(" Enter Trans Classes, in sequence or 0"),
              write("Tx Class #1:"), readint(Tx1),
              write("Tx Class #2:"), readint(Tx2),
              write("Tx Class #3:"), readint(Tx3),
              write("Tx Class #4:"), readint(Tx4),
              asserta(argn(Region,Tx1,Tx2,Tx3,Tx4)),
              asserta(runsin(Tx1,Region)), asserta(runsin(Tx2,Region)),
              asserta(runsin(Tx3,Region)), asserta(runsin(Tx4,Region)),
              asserta(runsat(Region,DPRTY)), closeall, morergn.
   morergn if makewindow(4,79,4,"Region",15,25,3,20), nl, nl,
              write(" Another RGN ? "), readln(Ans), Ans=y,
              removewindow, defrgn.
   morergn :- closeall.
```

Figure 4. Data Gathering and Assertion

```
chkindex(USIX,Access,Dbdname) :- USIX="y",
                changestatus(" Define Index pointing to Database"),
                lineinput_leave(16,40,27,27,27,"Enter Index Name: ",
                "index000",Ixn), frontstr(8,Ixn,Ixname,_),
                lineinput_leave(20,1,27,27,27,"Enter Blocksize: ",
                "4096",Blk), str_real(Blk,Block),
                lineinput_leave(20,40,27,27,27,"Is On Volume : ",
                "volser",Vol), Volume=Vol,
                asserta(indexfor(Ixname,Dbdname)),
                asserta(liveson(Ixname,Volume)),
                asserta(afile(Ixname,Access,Block)).

chkindex(USIX,Access,Dbdname) :- USIX="n".
```

Figure 5. Passing Parameters and NOP clauses

```
  setdevice(Device) :- Device="none".
  setdevice(Device) :- findall(Devs,dev_char(Devs,_,_,_),Devlist),
                       write("\n\n List of 'KNOWN ' DEVICES: \n\n"),
                       wrlist(Devlist),
                       changestatus(" Select DEVICE TYPE from Table above"),
                       write("\n\n Select DEVICE Type or enter
                       'DEF' to define a new one :"),
                       readln(Adev), getdev(Device,Adev), disks.

getdev(Device,Adev) :- Adev="DEF",
                       write("\n Name of New DEVICE: "), readln(Devname),
                       write("\n SEEK Time in Msecs: "), readln(Seek),
                       write("\n REVOLUTION Time in Msec: "), readln(Revtime),
                       write("\n DATA TRANSFER Rate in MB/Sec: "), readln(Transfer),
                       asserta(adevice(Device,Seek,Revtime,Transfer)).

getdev(Device,Adev) :- dev_char(Adev,Seek,Revtime,Transfer)
                       asserta(adevice(Device,Seek,Revtime,Transfer)).

getdev(Device,_) :-    write("\n The DEVICE was incorrectly entered
                       or not found. Retry: press any key..."), readchar(_), disks.

    wrlist(NL) :-      nl, write7 (NL,0), nl.
    write7(TL,7) :-    !, nl, write7 (TL,0).
    write7([H|T],N) :- write (H,"  "), N1=N+1, write7 (T,N1).
    write7([],_).
```

Figure 6. List Processing and Knowledge Base Searches

```
      .
      .
      .

, ims_log(LREC), MSIZ = (Insegm * Insize) + (Outsegm * Outsize),
                LOGREC = LREC * (MSIZ + Spasize) / 4096,
                append([LOGREC|_],List8,List9),
                sumlist(List9,Sum,Num), closeall,
                assertz(inclass(TXCLASS,Tlcname)),
                assertz(uses("CPU",Tlcname,Sum)),
                dodbd(IMSTYPE,Tlcname),
                moretlc(IMSTYPE,Tlcname).

append([],LL,LL).
append([New|L1], List2, [New|L3]) if append (L1,List2,L3).

sumlist([],0,0).
sumlist([H|T],Sum,Num) if sumlist(T,S1,N1) and Sum=H+S1,Num=1+N1.
```

Figure 7. Arithmetic and List Summarization

A PROLOG-based Expert System for Tuning MVS/XA

Dr. Bernard Domanski ***

The College of Staten Island / CUNY
Bell Communications Research, Inc.

ABSTRACT

This paper will discuss some of the issues involved in building an Expert System that embodies tuning rules for IBM's MVS/XA operating system. To understand the components of an Expert System and their functions, PROLOG on an IBM PC (Turbo-PROLOG from Borland International) was chosen as the development environment. The paper will begin by defining the key concepts about Expert Systems, Knowledge Engineering, and Knowledge Acquisition. The reader will be given a brief overview of PROLOG, from which we can explain how an inference mechanism was developed. Finally, the paper will describe the Expert System that was developed, and additionally will provide a set of key issues that should be addressed in the future. It is our overall objective to provide new insight into the application of AI to CPE.

1. Introduction

In his acceptance speech for the A. A. Michelson award in 1985, Dr. K. Mani Chandy said that it takes some ten to fifteen years for research in the universities to become practical technology for industry. The field of *Artificial Intelligence* (AI) is a prime example; the last fifteen years have seen AI flourish in coming from the university sector to industry. AI languages such as LISP and PROLOG have been touted as the languages of "choice" for fifth generation computing. The Japanese have chosen PROLOG as the machine language of their logic processor for their fifth generation systems.[1] Expert System shells such as OPS5[2] and M.1[3] can help simplify and speed the development of these systems.

Considering Computer Performance Evaluation (*CPE*), some prior exploratory work has been done.[4] [5] Vendors are now emerging with *Knowledge Based* systems for CPE (ISS, Boole & Babbage).

Although AI has had a long history, the application of AI to CPE is still immature. In CPE, many performance analysts have developed a set of "rules-of-thumb" for tuning large systems. Typically, this set takes five years to develop. As there is a tendency to promote quality personnel within a corporation, it is *critical* to extract the set of rules from the performance "expert" before (s)he is promoted to a new position. In essence, a "technology transfer" must occur.

This paper will discuss some of the issues involved in building an Expert System that embodies tuning rules for IBM's MVS/XA operating system. To understand the components of an Expert System and their functions, PROLOG on an IBM PC (Turbo-PROLOG from Borland International) was chosen as the development environment. The paper will begin by defining the key concepts about Expert Systems, Knowledge Engineering, and Knowledge Acquisition. The reader will be given a brief overview of PROLOG, from which we can explain how an inference mechanism was developed. Finally, the paper will describe the Expert System that was developed, and additionally will provide a set of key issues that should be addressed in the future. It is our overall objective to provide new insight into the application of AI to CPE.

2. What is an Expert System?

An *Expert System* is a knowledge based reasoning system that captures and replicates the problem solving ability of human experts. Expertise consists of knowledge about a particular area or *domain*, understanding of domain problems, and skill at solving some of these problems.[6]

Domain knowledge, rather than the complexity of formal reasoning methods, is the key to solving difficult problems. Knowledge can take many forms:

- Specific inferences that follow from specific observations,
- Necessary and sufficient conditions for achieving a goal,
- Probable causes of symptoms.

Knowledge can be represented in several ways, such as rules, frames, and logical predicates. In most current Expert Systems, knowledge is expressed in *rules*, and these *rule-based systems* will be discussed here. The skill of these systems increases at a rate proportional to the enlargement of their knowledge bases. They usually explain their conclusion by retracing their lines of reasoning and translating the logic of each rule used into natural language.

There are a variety of reasons for building systems that capture human expertise:

- Experts retire or are promoted, taking their knowledge with them.
- An expert's time is not particularly well spent answering users' questions.
- Expertise may be scarce.
- Expertise may be expensive to deliver.
- Experts are not always consistent.

Figure 1 shows a diagram of the most basic components of an expert system. The *knowledge base* is usually a collection of rules about the problem domain, supplied by the expert. Problem solving knowledge is usually used in consultations between the Expert System and the end user or with other programs. The *working memory* keeps track of what the system knows during each consultation. The *inference engine* looks in the working memory and the knowledge base to see what is true at any given time to resolve queries.

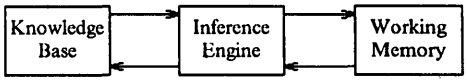


Figure 1

Consider a simple knowledge base with only 2 rules:

1. Rule 1: IF *a* THEN *b*.
2. Rule 2: IF *b* THEN *c*.

Each rule has a *premise* clause (IF), and a *conclusion* clause (THEN). If a user poses the question "IS *c* TRUE", the inference engine looks through **all** the rules

*** The author is currently on a sabbatical leave from the College of Staten Island, a unit of the City University of New York. This work was conceived and performed while at Bell Communications Research, Inc., Piscataway, New Jersey.

1. Feigenbaum, E. A., McCorduck, P., "The 5th Generation - Artificial Intelligence and Japan's Computer Challenge to the World", Addison-Wesley, 1984.
2. Forgy, C. L., OPS5 User Manual, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA., 1981.
3. M.1 is a product of Teknowledge, Inc., Palo Alto, CA.
4. Hellerstein, J., Van Woerkom, H., "YSCOPE: A Shell for Building Expert Systems for Solving Computer Performance Problems", Proceedings, Computer Measurement Group, Dallas, December, 1985.
5. Artis, H. P., "Using Expert Systems for Analyzing RMF Data",ibid.
6. Hayes-Roth, "Rule Based Systems", Communications of the ACM, 28(9), 1985.

in the knowledge base and tries to find one that would conclude about *c*. It finds rule 2: IF *b* is true, THEN *c* is true. Now, if the inference engine could discover whether *b* was true, then it would know whether *c* was true. So a query is made of working memory: "Is *b* true?" Again, the inference engine examines the rules in the knowledge base, this time looking for a rule that would come to a conclusion about *b*. It finds rule 1 - this leads to asking the question "Is *a* true?" If the user responded "yes", then the inference engine would:

- erase the query about *a*,
- place the fact that *a* is true in the working memory,
- use Rule 1,
- place the fact that *b* is true in the working memory,
- use Rule 2,
- place the fact that *c* is true in the working memory,
- resolve the original query, and
- tell the user "Yes, *c* is true."

This is a simple example of a **backward chaining** system. The inference engine works backwards through the conclusions of the rules, trying to determine whether the premises are true. In a **forward chaining** system, facts would be placed in the working memory and the inference engine would report any conclusions it could to the user. For instance, if "*a* is true" were entered, the system could report that both *b* and *c* were true. Forward chaining was used in our PC/PROLOG Expert System for MVS/XA. Backward chaining systems tend to be *goal driven* - they work backward from some specific goal (*c* above). Forward chaining systems tend to be *data driven* - the system is primed with certain facts and draws any conclusions it can.

Other information must reside in the knowledge base in addition to the rules. For example, consider the user interface. How does the Expert System know what question to ask the user when that becomes necessary? This information is stored in the part of the knowledge base that describes the *a*'s and the *b*'s - the **traits** characteristic of a given problem domain. For example, the trait USERS may carry with it the prompt "On average, how many users are connected to the system?"

But when many rules are present in the knowledge base, complex consultations can occur. It can become difficult to predict what questions will be asked and in what order unless some precautions are taken. The builders of the famous MYCIN medical Expert System[7] found that doctors were somewhat bewildered when the consultation questions jumped from one topic to another: "What is the patient's age?" "What was the result of test X?" "What is the patient's name?" and so on. It becomes necessary to use more control information. For instance, the system could be instructed "Ask about traits *a*, *b*, and *c* first, then try to find the value of trait *d* by forward chaining, then ask about traits *e* and *f*, ..." This consultation and control information is also kept in the knowledge base. As will be shown later, the PC/PROLOG Expert System for MVS/XA employed this concept in its structure of its knowledge base.

It is sometimes necessary to keep even more information in the knowledge base, such as special purpose algorithms, or methods needed to retrieve information from databases that reside outside the Expert System. Thus the task of **knowledge engineering** includes more than eliciting rules from the expert; all the traits (such as *a* and *b* above), and their characteristics and accompanying control information must be identified as well.

A natural question asked about Expert Systems is "what are the advantages of using a knowledge based system?" In general, knowledge based systems extend the speed, accuracy, consistency, availability, and affordability of computation from *clerical* to *intellectual* applications. Specifically,

- *Behavior* - Users can ask "Why?" questions to show why certain decisions were made. This is in direct contrast to a traditional data processing system, where the only way to verify the system's reasoning is to simulate the system's behavior.
- *Knowledge Independence* - A piece of knowledge (a rule) can be examined and easily changed somewhat independently from other pieces of knowledge in the system.
- *Separation* - Knowledge is kept separate from the reasoning process that uses it. Building an Expert System consists of changing *knowledge* rather than program code. The expert and the knowledge engineer can examine and change this knowledge since it is ideally represented in an English-like format. Because of this separation, rule based systems can be incrementally developed with steady performance improvements.

7. Buchanan, B. G., Shortliffe, E. H., "Rule-Based Expert Systems: The MYCIN Experiments of the Heuristic Programming Project", Addison-Wesley, Menlo Park, CA., 1984.

3. Knowledge Engineering / Knowledge Acquisition

What is Knowledge Engineering? This is the term given to the extraction, description, and computerization of expert knowledge. Knowledge consists of descriptions, relationships and procedures in a domain of interest. The major task in building an Expert System is the transfer of knowledge from its source into the Expert System; this is called **knowledge acquisition.**

Potential sources of knowledge include human experts, textbooks, data bases, examples, case studies, and personal experience. Within CPE, nearly all these sources can be used to elicit performance knowledge. The prototype system developed using PROLOG used knowledge that was extracted from technical papers, IBM manuals, and personal experience.

The process of building a comprehensive Expert System can be described as an iterative, cyclic interaction between the expert, the knowledge engineer, and the system itself:

1. The expert tells the knowledge engineer what rules to add or change.
2. The knowledge engineer makes changes to the knowledge base.
3. Several tests are run for consistency checking.
4. If any problems arise, they are discussed with the expert, and we go back to step 1.
5. The expert runs the modified system on new cases until new problems are discovered.
6. If no problems are encountered in a substantial number of cases, the expert stops checking the system.

Hayes-Roth outlines stages of Expert System knowledge acquisition that roughly correspond to systems analysis and software engineering. In this acquisition model, an Expert System grows by proceeding from simple to increasingly more difficult tasks and by incrementally improving the representation and organization of the knowledge. We outline this acquisition model below.

- *Identification* - Identify the problem area and define its scope while determining the resources needed and deciding on the goals and objectives of the system. Questions that must be answered include "What class of problems will the Expert System be expected to solve?", "What data is required?" and "What are important terms and their interrelationships?"
- *Structure* - Key concepts, relations, and information-flow characteristics needed to describe the problem-solving process are defined. Related questions might include "What is given and what is inferred?", "How are objects related?", and "What is the information flow?"
- *Formalization* - Concepts and relations are mapped into a formal structure suggested by the Expert System. Related questions include "What kinds of questions should be asked to obtain data?", or "What is the cost of data acquisition?"
- *Implementation* - A knowledge engineer combines and organizes the formalized knowledge to make it compatible with the information flow characteristics of the problem, resulting in a prototype that can be executed and tested.
- *Testing* - The performance of the prototype is evaluated and revised to conform to the standard defined by experts in the problem domain. Typically, this is an iterative process: an expert evaluates the program's performance, while the knowledge engineer revises the knowledge base.

It is beyond the scope of this paper to examine knowledge acquisition techniques in detail. Note that most of these techniques require that knowledge be elicited from information sources and first placed into an **information base**. From there, it is analyzed and organized into knowledge bases, and then used with an Expert System. The knowledge must be tested for *necessity and sufficiency*, and the knowledge bases must be refined accordingly. Techniques in knowledge acquisition often have roots in psychology: the Delphi technique[8] and Personal Construct Theory.[9] The interested reader is encouraged to examine

8. Jagannathan, V., Elmaghraby, A. S., "MEDKAT: Multiple Expert Delphi-Based Knowledge Acquisition Tool", Computer Science Department Technical Report, University of Louisville 1985.
9. Boose, J. H., "Personal Construct Theory and the Transfer of Human Expertise", Proceedings National Conference on Artificial Intelligence, Austin, Texas, 1984.

these and other sources of knowledge acquisition information.

Our PC/PROLOG based Expert System for tuning MVS/XA faced the same problem of knowledge engineering / knowledge acquisition. Though we chose as our initial knowledge source written material (e.g. papers, IBM manuals), we first addressed questions of identification. We felt that the most common types of tuning problems for MVS/XA are I/O related; thus, the class of problems for our first prototype was limited to I/O and paging/swapping. We next needed to identify the data that would be required, and this was inferred from our information sources. For example, we decided that it would be important to know whether or not dedicated page volumes were being used. Overall, the objective of our system was to gather information about the presence/absence of performance "symptoms", analyze these to form a "diagnosis", and then suggest a "treatment."

In defining the structure, we grouped certain traits together that are related to specific problems. For instance, values for pend time, connect time, disconnect time and IOS queue time are related to poor I/O response time - we could derive a value for I/O response time from these traits. The procedures for finding a problem could then be informally described by grouping the key traits together, inferring any additional information, and finally describing (informally) the relationships (rules) between the traits and the problems to be solved.

The formalization of structure into a more formal representation is usually made easy when an Expert System shell is used. The format of rules is specified by the shell, so the transformation does not have to be difficult. In our case, no such shell was used; our problem was compounded because we were developing our own shell. In a later section of the paper, we will describe the steps that were taken to build a shell and to determine a "language" for expressing CPE rules.

Implementation implies that the system be built. This process will be described later. But an interesting question about the focus of rules comes up. We might build a rule like "If page delay time is greater than 100 msec, then you should examine the page/swap dataset configuration." Here, the presence of the symptom (page delay time > 100 msec) implies a potential diagnosis, which would have a corresponding treatment (e.g. consider dedicated page volumes). We will call this a *forward leading predicate*. But now consider the following: if a treatment was "use dedicated page volumes", then couldn't the Expert System ask "are you using dedicated page volumes?" If the answer were yes, then suggesting this as a possible treatment is redundant and confusing. This is an example of a *backward looking predicate* - the presence of a potential treatment would imply that the Expert System should look for *other* treatments, or, reply that the system does not know a treatment for the problem.

Testing our prototype consists simply of letting an "expert" use the system, and incorporating any additional knowledge the expert gives into the system. For this, our prototype was given a "why" facility - the ability for the user to ask how a particular diagnosis was reached. In addition, the prototype was given an editing capability over the contents of the knowledge base. Using commonly used PC editing keys, the expert could **add, delete or modify** not only the rules in the system, but the control mechanism, and the contents of the help, diagnosis and treatment messages that the system uses. In short, the expert is given full control of all of the knowledge that the system has during the entire testing process. After testing, these features were left in the Expert System.

4. PROLOG

Over the past two decades, the price of hardware has dropped dramatically, while the costs associated with software development have now become the dominant portion of a total system budget. This rapid rise has influenced the development of new programming tools that simplify system development. PROLOG is the result of years of effort in this area. PROLOG was developed at the University of Marseilles, France by Alain Colmerauer in the early 1970's. It was designed to be a convenient tool for *PRO*gramming in *LOG*ic (hence the name). It can be more powerful and efficient than many traditional programming languages (e.g. PASCAL, FORTRAN, etc.)

A PROLOG program uses a description of a problems' facts and rules, and then finds all possible solutions to the problem. In PROLOG, a programmer describes *what* must be computed, rather than how the computation should be carried out. This declarative (rather than procedural) approach eliminates well known errors that are common in other programming languages (e.g. one too few iterations in a loop). So aside from some initial declarations, a PROLOG program consists of a set of *facts* (e.g. john likes mary, tom likes sam), and a set of *rules* (e.g. jean likes X if tom likes X). PROLOG *deduces* that jean likes sam. Goals like "find every person who likes sam" can be asked of a PROLOG program. PROLOG uses a built-in backtracking mechanism that, once one solution has been found, causes PROLOG to reevaluate any assumptions that were made to see if some new variable values will provide additional solutions.

To better understand the syntax and operation of PROLOG, consider the following sample program:

```
clauses
   watches(bill,bob).
   watches(john,jane).
   watches(fred,felicia).
   watches(mitch,bill).
   watches(brenda,greg).
   watches(bob,bob).
   watches(fred,greg).
   watches(bill,phillipe).
```

The clause "watches(bill,bob)" is the fact "bill watches bob." Thus, if we pose as a goal "watches(mitch,bill)", PROLOG would reply "true". "watches(bill,jane)" would result in a reply of "false". Variables begin with upper case characters; thus a goal of "watches(john,Who)" (analogous to asking Who does john watch?) causes a reply of "jane". But the goal of "watches(fred, Who)" results in **both** "greg" and "felicia". The "_" character is called the blank variable; a goal of "watches(fred,_)" is analogous to asking "does fred watch anyone?" This would result in the reply true.

If we add the *rule* "happy(eric) if watches(brenda,eric)" (i.e. eric is happy if brenda watches eric), the goal "happy(eric)" ("is eric happy") would be satisfied. The rule "nervous(Who) if watches(bill,Who)" would be satisfied with "bob" and "phillipe"; this is analogous to asking "Who is nervous", and answering using the rule "someone is nervous if bill is watching them." Thus, PROLOG looks through the already established facts and concludes that since bill watches both bob and phillipe, that both bob and phillipe are nervous.

Though there are many interesting features of the language (the reader is encouraged to examine the cut operator, as well as backtracking), the most commonly used *data structure* is known as a **list.** Analogous to arrays, lists are collections of objects (elements) of the same type, separated by commas and placed inside square brackets. For example, **[1, 2, 3]** is a list of the integers 1, 2, and 3. PROLOG can manipulate a list by dividing it into two parts: a *head* and a *tail.* The head is the first object on the left in the list; the tail is the rest of the list. Syntactically, a PROLOG list is written with a vertical bar separating the head from the tail **[head | tail]**. Lists are treated just like other objects in PROLOG, and PROLOG has built-in facilities (predicates) for manipulating the elements of a list.

5. The Development of the Expert System

The reader should, as this author did, feel comfortable with the concept of an Expert System, as well as some PROLOG basics. The discussion now turns to building the system itself. Consider that this author is (by definition) a programmer of *procedural* languages (FORTRAN, PL/I, C, etc.) Thus, PROLOG is somewhat unnatural, and causes some conceptual problems that have to be overcome. For example, consider the following rule: Bad I/O response time exists if pend time is greater than or equal to 10% of the sum of disconnect plus connect time, and IOS queue time is greater than the sum of pend time, disconnect time and connect time. In PROLOG,

```
bad_resp(IOSQ, PEND, DISC, CONN) if
   PEND > = (DISC + CONN) * 0.10  and
   IOSQ > (PEND + DISC + CONN).
```

This simple rule implies the our PROLOG program should:

- Prompt the user for the value of each variable (IOSQ, PEND, etc.),
- Provide help with each prompt should the user not understand what is being asked for,
- Validate each resulting reply - here, with a simple numeric test,
- Pass the replies as parameters to this rule that does the evaluation (bad_resp(IOSQ, PEND, DISC, CONN)),
- If the rule is satisfied, provide the user with the diagnosis (poor I/O response time because of these symptoms),
- Look up a treatment for this diagnosis.

Our first version of the system has the rule (knowledge) coded in PROLOG; thus it is part of the reasoning process. Given all the rules to be coded, plus the help, diagnosis and treatment messages, the resulting program was rather long, and strongly resembled a traditional procedural program.

Recall that an objective of an Expert System is to *separate* the knowledge from the reasoning process. Our first version mixed the knowledge and reasoning process together. Thus, we examined the rules a little closer, since they were the driving force behind the design of the program. When invoked, rules result in true or false. Thus, if the prompts were structured as true or false

questions, the inference mechanism could be greatly simplified. For example, consider the following:

```
demand_paging_problem(P1, P2) if
  ask(i1, P1) and
  ask(i2, P2).

ask(Index, Prompt) if
  prompt(Index, Prompt) and
  write("Is ",Prompt,"? ") and   /* an ordinary write   */
  readln(Reply) and              /* read the user reply */
  yes(Reply).                    /* validate the reply  */

prompt(i1,"Demand Paging High").
prompt(i2,"storage isolation used for IMS or CICS").
```

The inference mechanism "ask" is now divorced from the knowledge of what prompts should be used. The prompts themselves are associated with specific *traits* (i1, i2). For illustration, we have included the prompt clauses within the program, but the collection of prompt clauses is ordinarily kept within a *separate, editable file.*

Note though that the control information - ask about trait i1 then trait i2 relative to demand paging - is still part of the PROLOG program. Thus, we need to carry this "separation" of knowledge from reasoning one step further. We create in a separate, editable file, a **knowledge base** of the following form:

- prompt(trait, prompt messages).
- condition(symptom, [list of traits]).
- diagnosis(symptom, [list of traits], treatment).
- treatmsg(treatment, treatment messages).
- diagmsg (treatment, diagnosis messages).
- helpmsg (trait, help messages).

where:

- *prompt* - associates with each trait the corresponding prompt message to ask the user. "Is" is printed before each prompt message implying a true/false question.
- *condition* - associates a list of traits with a particular symptom. For example, condition(ratehi, [i1, i2, i3]) would imply that traits i1, i2 and i3 are all associated with the symptom ratehi. This knowledge is used to *control* the flow of prompting.
- *diagnosis* - associates the true/false values of traits with whether a particular symptom exists. If the symptom does exist, *treatment* is the index to invoking the correct diagnosis and treatment messages. For example, consider
  - diagnosis(ratehi, [i1, i3], fixrate).
  - prompt(i1, "Demand Paging High").
  - prompt(i3, "Storage Isolation Used for IMS or CICS").

  This implies that the condition ratehi is present if traits i1 and i3 are both true - demand paging is high and storage isolation is being used for either IMS or CICS. A minus sign "-" in front of a trait, e.g. -i2, would imply that trait i2 is tested for falsehood. On closer examination, *diagnosis* represents how tuning rules are coded. If it is possible for a symptom to be caused several different ways, then each possibility can be coded; the inference mechanism automatically checks for all possible solutions.
- *treatmsg, diagmsg* - associate a treatment with a set of messages - treatmsg deals with treatments, diagmsg deals with diagnoses.
- *helpmsg* - associates a trait with a set of help messages that try to give more information about the particular trait.

Thus, each of the above represents knowledge, and is kept in a file that can be edited by the user. The inferencing mechanism is thus reduced to processing these predicates; it is short, and contains no knowledge. Conceptually, it can be used in any expert application as long as the syntax outlined above is followed.

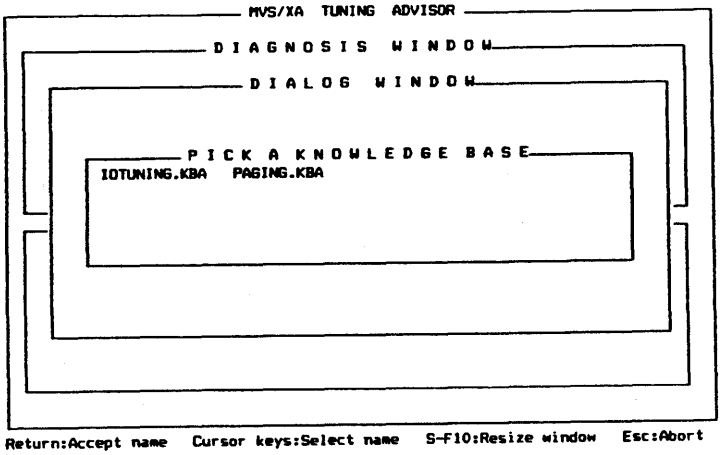
Several features were added to the general inferencing mechanism: after a consultation, the user can invoke a built-in editor, which resembles the PC word processing program WORDSTAR.[10] Rules can be added, deleted or modified. Care should be taken so as not to violate the syntax outlined above - no syntax checking is done. In addition, a "Why?" feature was added. After each diagnosis is given, the user is given the opportunity to ask the Expert System why that diagnosis is being given; in other words, how that conclusion was reached. The inferencing mechanism *backtracks* from the treatment using the diagnosis and prompt predicates to replay the rules that were invoked using English.

6. A Consultation Session

It is difficult for the reader to get a firm understanding of the Expert System described unless (s)he has the actual floppy disk with the program available. To eliminate this difficulty, this section will display parts of a sample consultation session with the Expert System. The discussion will show the screen displays a user would see, and will provide an overview of the overall flow of the system. Finally, we will give a short discussion of how a knowledge base can be edited.

Required Files - First, the program (henceforth called *ADVISOR*) must be present along with at least one knowledge base. Knowledge bases are ordinary text files that can be created by any ordinary PC editor that does not use embedded control characters. *ADVISOR* knows that files are knowledge bases if they end with the suffix KBA (for *K*nowledge *BA*se). In our prototype, we created two knowledge bases, IOTUNING and PAGING.

The following is what is displayed when *ADVISOR* is invoked:



The user uses the PC arrow keys to select which knowledge base they want to use; in essence, this is actually equivalent to selecting a *knowledge domain* to be exploited.

At this point, we point out the different *windows* that *ADVISOR* uses.

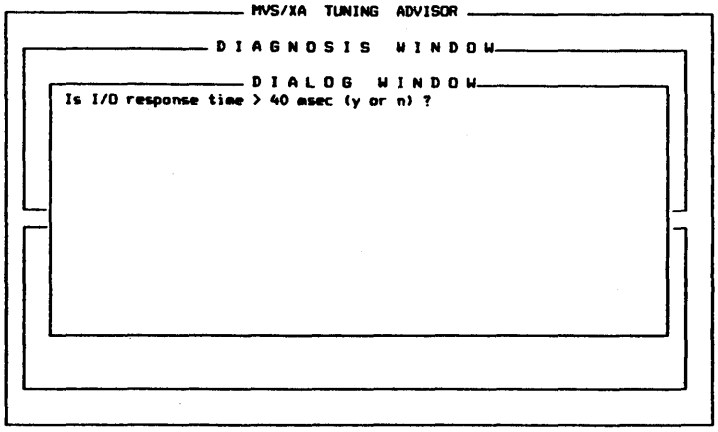
- *DIALOG* - This window is the primary means of communication between *ADVISOR* and the user. Performance questions are posed to the user, and each reply should be followed by hitting the RETURN key.
- *HELP* - Should a user need assistance in understanding a question, the HELP window will display that information. HELP is invoked by replying with a ? to the question posed in the DIALOG window. In addition, the HELP window is used later to explain why a particular diagnosis was arrived at.
- *DIAGNOSIS* - After a set of related performance questions is asked, any performance problem found will be noted in the DIAGNOSIS window. Here, the user is given the opportunity of asking why a particular DIAGNOSIS is being made.

---

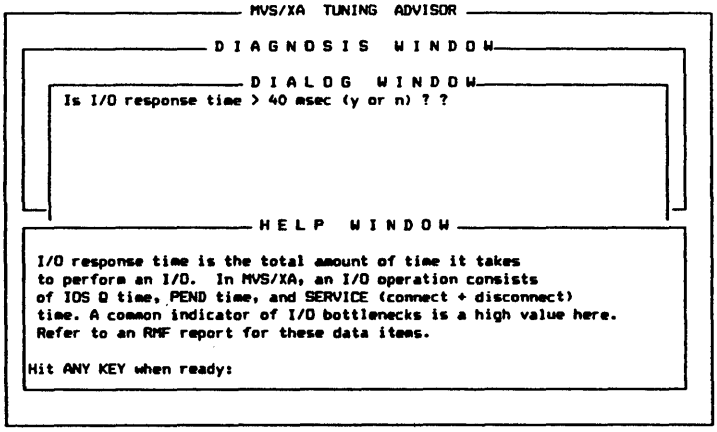
10. WORDSTAR is a registered trademark and product of the MicroPro International Corp.

- *TREATMENT* - Often, a particular diagnosis will have one or more possible remedies, or TREATMENTS. Each possible TREATMENT is displayed in this window successively, each one being preceeded by giving the user the opportunity to ask WHY.
- *EDIT* - Once a consultation session has ended, the user is given the opportunity to edit a knowledge base. Here, a *full-screen* editor is invoked that allows the rules in the knowledge base to be modified, deleted or extended.

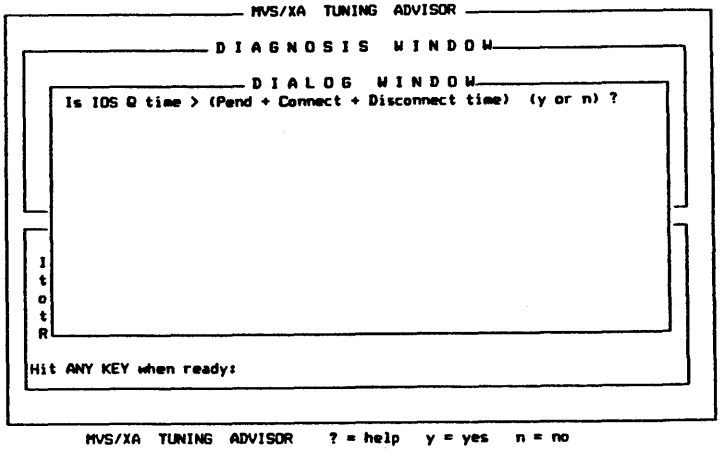
We now examine several screens to become familiar with the "look and feel" of the system. As each question is asked in the DIALOG window -


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
DIALOG WINDOW
Is I/O response time > 40 msec (y or n) ?
MVS/XA TUNING ADVISOR ? = help y = yes n = no

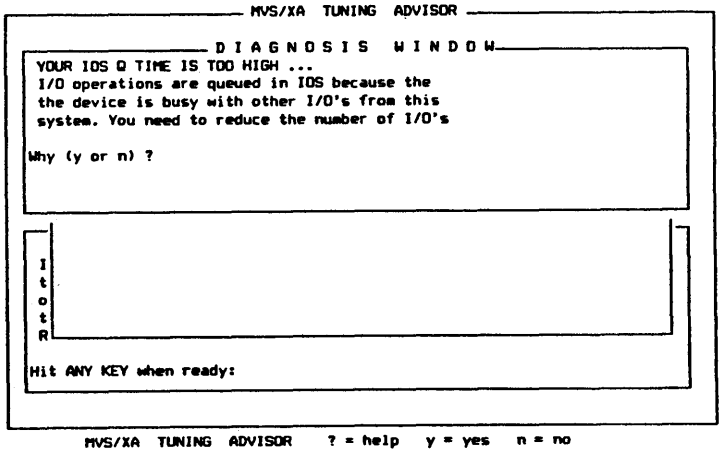
the user can ask for help by typing ? and hitting the return key -


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
DIALOG WINDOW
Is I/O response time > 40 msec (y or n) ? ?
HELP WINDOW
I/O response time is the total amount of time it takes
to perform an I/O. In MVS/XA, an I/O operation consists
of IOS Q time, PEND time, and SERVICE (connect + disconnect)
time. A common indicator of I/O bottlenecks is a high value here.
Refer to an RMF report for these data items.
Hit ANY KEY when ready:
MVS/XA TUNING ADVISOR ? = help y = yes n = no

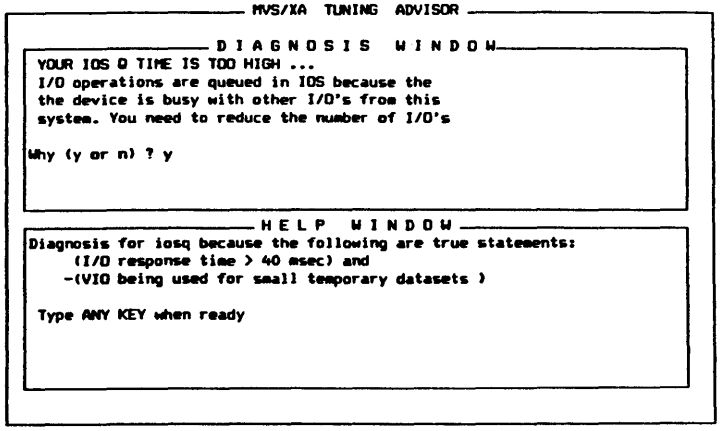
When finished with help, typing any key will return the user to the DIALOG window, where (s)he is reminded to reply with a *y* (yes) or an *n* (no). From there, the next question is asked -


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
DIALOG WINDOW
Is IOS Q time > (Pend + Connect + Disconnect time) (y or n) ?
Hit ANY KEY when ready:
MVS/XA TUNING ADVISOR ? = help y = yes n = no

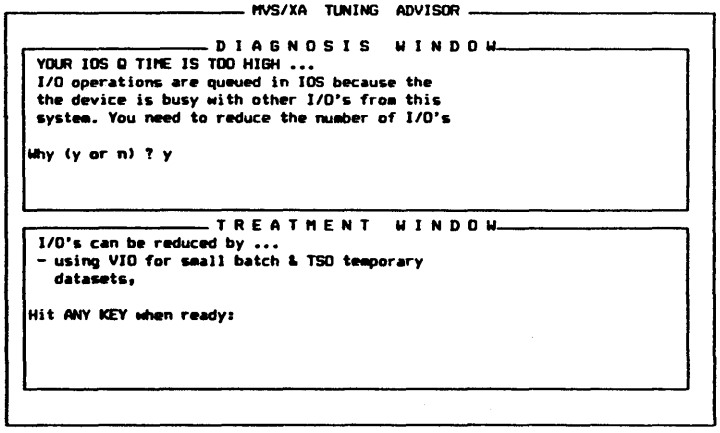
Once all of the questions or *traits* corresponding to a logical sequence have been asked, descriptions of any performance problems detected appear in the DIAGNOSIS window -


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
YOUR IOS Q TIME IS TOO HIGH ...
I/O operations are queued in IOS because the
the device is busy with other I/O's from this
system. You need to reduce the number of I/O's
Why (y or n) ?
Hit ANY KEY when ready:
MVS/XA TUNING ADVISOR ? = help y = yes n = no

Here, the user is given the opportunity to ask what caused (WHY) this diagnosis to be made by *ADVISOR*. If the user replies *n* to the WHY prompt, the TREATMENT window appears (more later). If the user replies *y* to the WHY prompt, the HELP window reappears containing an English-like replay of the prompts used to arrive at the current diagnosis -

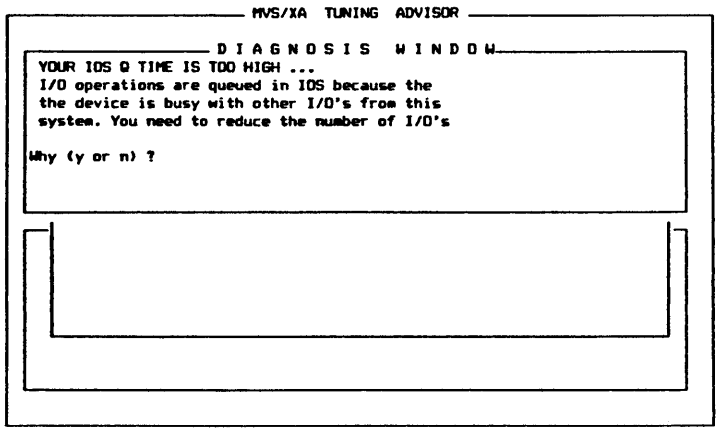

MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
YOUR IOS Q TIME IS TOO HIGH ...
I/O operations are queued in IOS because the
the device is busy with other I/O's from this
system. You need to reduce the number of I/O's
Why (y or n) ? y
HELP WINDOW
Diagnosis for iosq because the following are true statements:
(I/O response time > 40 msec) and
-(VIO being used for small temporary datasets)
Type ANY KEY when ready
MVS/XA TUNING ADVISOR ? = help y = yes n = no

Notice, if a particular prompt is preceeded with a hyphen (-), this implies the *negation* of the particular prompt. Next, the user types any key which causes the HELP window to be replaced by the TREATMENT window. Here, a possible suggestion is made that the user can adopt to ultimately help tune the actual MVS system.

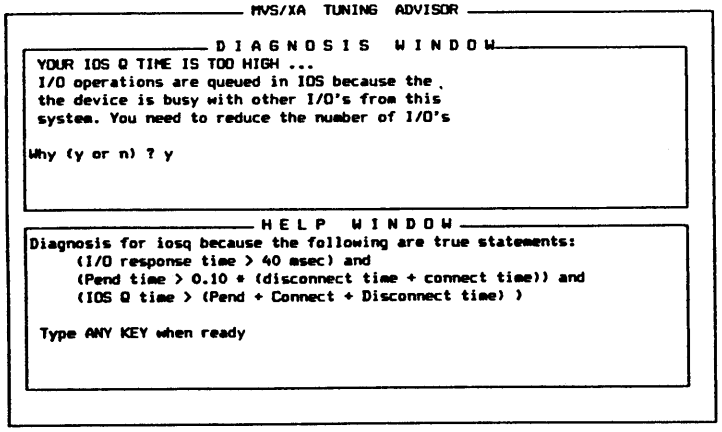

MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
YOUR IOS Q TIME IS TOO HIGH ...
I/O operations are queued in IOS because the
the device is busy with other I/O's from this
system. You need to reduce the number of I/O's
Why (y or n) ? y
TREATMENT WINDOW
I/O's can be reduced by ...
- using VIO for small batch & TSO temporary
datasets,
Hit ANY KEY when ready:
MVS/XA TUNING ADVISOR ? = help y = yes n = no

As before, the user types any key which causes *ADVISOR* to continue.

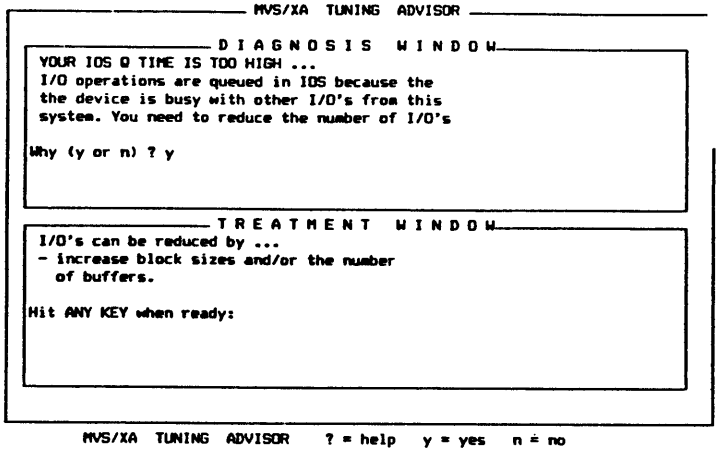
If a particular diagnosis reappears following the disappearance of the TREATMENT window,


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
YOUR IOS Q TIME IS TOO HIGH ...
I/O operations are queued in IOS because the
the device is busy with other I/O's from this
system. You need to reduce the number of I/O's
Why (y or n) ?
MVS/XA TUNING ADVISOR ? = help y = yes n = no

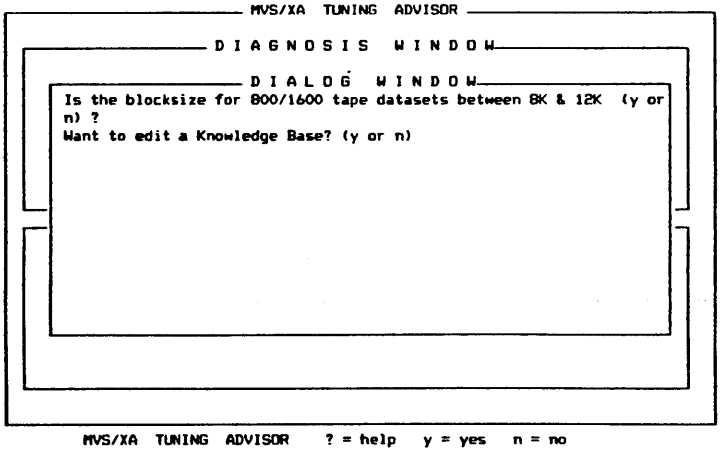
it implies that the current diagnosis has been reached by invoking additional rules. Again, the user can ask WHY, and we can see that a different reply is given in the HELP window.


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
YOUR IOS Q TIME IS TOO HIGH ...
I/O operations are queued in IOS because the
the device is busy with other I/O's from this
system. You need to reduce the number of I/O's
Why (y or n) ? y
HELP WINDOW
Diagnosis for iosq because the following are true statements:
(I/O response time > 40 msec) and
(Pend time > 0.10 * (disconnect time + connect time)) and
(IOS Q time > (Pend + Connect + Disconnect time))
Type ANY KEY when ready
MVS/XA TUNING ADVISOR ? = help y = yes n = no

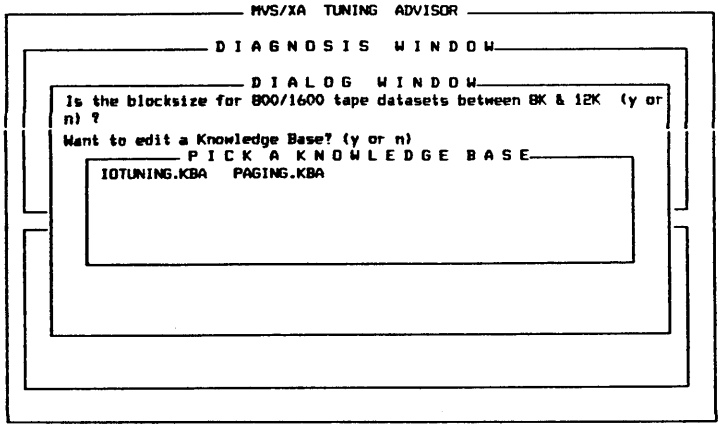
And, as before, the HELP window is replaced by the TREATMENT window, this time giving a *different* remedy the user might apply to the actual MVS system.


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
YOUR IOS Q TIME IS TOO HIGH ...
I/O operations are queued in IOS because the
the device is busy with other I/O's from this
system. You need to reduce the number of I/O's
Why (y or n) ? y
TREATMENT WINDOW
I/O's can be reduced by ...
- increase block sizes and/or the number
of buffers.
Hit ANY KEY when ready:
MVS/XA TUNING ADVISOR ? = help y = yes n = no

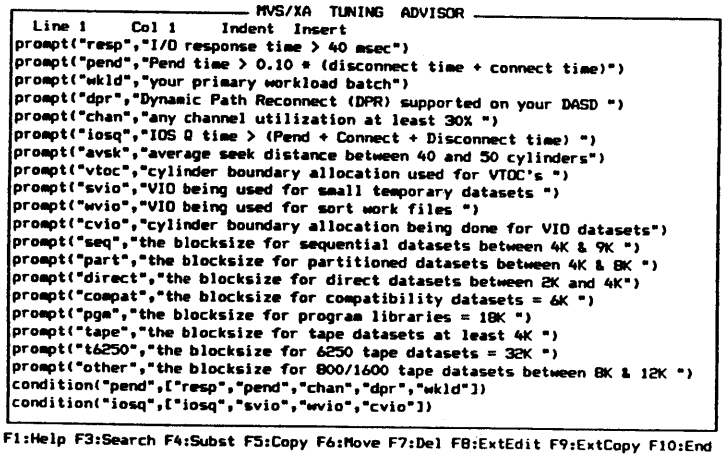
Once the consultation session has ended, the user is given the opportunity to edit a knowledge base - (a no reply ends the session).


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
DIALOG WINDOW
Is the blocksize for 800/1600 tape datasets between 8K & 12K (y or n) ?
Want to edit a Knowledge Base? (y or n)
MVS/XA TUNING ADVISOR ? = help y = yes n = no

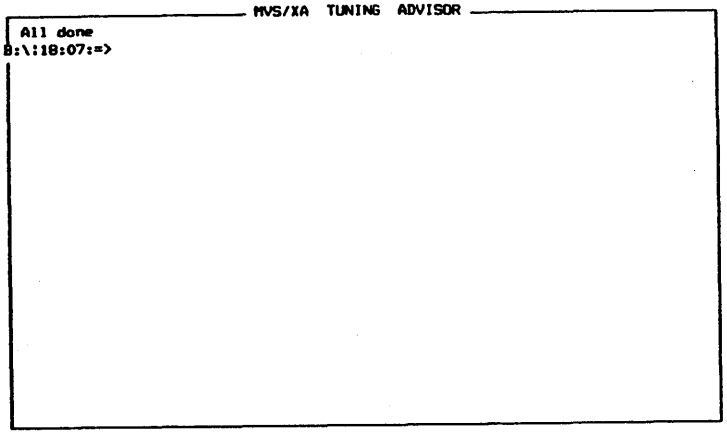
As at the start of program execution, the user selects a knowledge base with the PC arrow keys followed by enter. Here, a full-screen editor similar to WORDSTAR is invoked that operates on the knowledge base.


MVS/XA TUNING ADVISOR
DIAGNOSIS WINDOW
DIALOG WINDOW
Is the blocksize for 800/1600 tape datasets between 8K & 12K (y or n) ?
Want to edit a Knowledge Base? (y or n)
PICK A KNOWLEDGE BASE
IOTUNING.KBA PAGING.KBA
Return:Accept name Cursor keys:Select name S-F10:Resize window Esc:Abort

The arrow keys, PgUp, and PgDn are used to move around the window, the Ins and Del keys are used to set/unset insert mode and for character deletion (respectively). Help on all the editing functions is available by hitting the F1 key. The editor has search (F3), substitute (F4), copy (F5), move (F6), and block delete (F7) capabilities. F8 is an *auxiliary* edit function. When invoked, the user can edit yet another file while remaining within the edit session for the knowledge base. This is similar to the split-screen facility under TSO/ISPF on the mainframe. F9 is an external copy function. Another file (e.g. a knowledge base previously created) can be copied in part or as a whole to any section of the knowledge base being edited. F10 is used to end the editing session; here, the user is asked if (s)he wants to save the file just edited.


MVS/XA TUNING ADVISOR
Line 1 Col 1 Indent Insert
prompt("resp","I/O response time > 40 msec")
prompt("pend","Pend time > 0.10 * (disconnect time + connect time)")
prompt("wkld","your primary workload batch")
prompt("dpr","Dynamic Path Reconnect (DPR) supported on your DASD ")
prompt("chan","any channel utilization at least 30% ")
prompt("iosq","IOS Q time > (Pend + Connect + Disconnect time) ")
prompt("avsk","average seek distance between 40 and 50 cylinders")
prompt("vtoc","cylinder boundary allocation used for VTOC's ")
prompt("svio","VIO being used for small temporary datasets ")
prompt("wvio","VIO being used for sort work files ")
prompt("cvio","cylinder boundary allocation being done for VIO datasets")
prompt("seq","the blocksize for sequential datasets between 4K & 9K ")
prompt("part","the blocksize for partitioned datasets between 4K & 8K ")
prompt("direct","the blocksize for direct datasets between 2K and 4K")
prompt("compat","the blocksize for compatibility datasets = 6K ")
prompt("pgm","the blocksize for program libraries = 18K ")
prompt("tape","the blocksize for tape datasets at least 4K ")
prompt("t6250","the blocksize for 6250 tape datasets = 32K ")
prompt("other","the blocksize for 800/1600 tape datasets between 8K & 12K ")
condition("pend",["resp","pend","chan","dpr","wkld"])
condition("iosq",["iosq","svio","wvio","cvio"])
F1:Help F3:Search F4:Subst F5:Copy F6:Move F7:Del F8:ExtEdit F9:ExtCopy F10:End

Finally, the session ends.



7. Issues for Future Development

As this PC/PROLOG based system is still in its infancy, several issues regarding where the system should next go need to be expressed. The following issues should apply to any Expert System for CPE.

1. *Location Help* - Rather than just explaining the meaning of traits in help messages, add information about where to find measurement(s) associated with the trait; e.g. on page 2, bottom of an RMF report. This is an example of using an additional class of knowledge that, until now, only the expert has.
2. *Automatic Data Reduction* - Ideally, a "front-end" mechanism that could read and analyze measurement sources like RMF automatically would greatly simplify the user's interaction with the system. Rather than putting the burden of responsibility on the user to get the answers to the prompts, a program(s) could be designed to extract the corresponding answers, and download these to the PC. A potential source for much of this data could be the MICS data base (Morino Associates), which is a repository for many different types of measurements. The MICS data dictionary already assigns names to many measurements, and these names could be related to the traits we referred to previously.
3. *Links to other PC packages* - If data were downloaded from the mainframe to the PC, it could be stored in a spreadsheet or a database. Products like 1-2-3 from Lotus Development or dBASE-III from Ashton-Tate could then be used to do the analysis on the PC rather than the mainframe. These could be invoked from PROLOG directly, as the interface is already defined.
4. *Certainty Values* - As Artis pointed out (5), some rules are not as clear cut as one would like. If a certainty value, say between 0 and 100, was attached to each rules' conclusion, a *certainty threshold* could be specified for the premises. Below this threshold, rules are considered to be false. This would allow *fuzzy reasoning* to be a part of the system.
5. *External Program Execution* - As in many mainframe offerings, it is sometimes convenient to have an **exit** facility. This would allow the PROLOG system to escape to execute some user-specified program, and then return. This could greatly increase the potential applicability of the system.

8. Conclusions

The objective of this paper was to provide new insight into CPE by applying AI techniques. An Expert System is a complicated object with many advantages; foremost among these is that expert knowledge is elicited and stored in an executable form.

The system described is only a beginning at applying current technology. It was **never intended** to replace the human expert. On the contrary, the original intent was to use the system as a training mechanism, where a new CPE practitioner could pose performance scenarios to the Expert System, and see how these would be analyzed, and what recommendations would be made.

It is our belief that similar applications will flourish in the next decade. Yet, though the size of knowledge bases may increase, we believe that the human inference mechanism is still better than any that will be built during that time. We conclude with the following from Aristotle: "It is the mark of an instructed mind to rest satisfied with the degree of precision which the nature of the subject admits, and not to seek exactness when only an approximation of the truth is possible."

9. Acknowledgements

The author would like to take this opportunity to thank those referees and Editorial Review board members of the Computer Measurement Group for their constructive comments and suggestions. In addition, we would like to thank Bell Communications Research for the opportunity to explore this area. In particular, the members of Bellcore's Performance Management District served as a *set* of experts from which the Expert System was examined. Their input is gratefully appreciated.

A Flexible Approach to Performance Prediction For Large System Software Development

by A. Long and W. Suhler
IBM Corporation

ABSTRACT

Assessing large system software performance is a difficult task at best. When much of the system's software is still being designed and coded, the rapidly changing environment of the project increases the difficulty of the assessment task. In this situation a performance evaluation system must have the capability to meet the wide-ranging demands for performance analysis and the flexibility to handle the "moving target" nature of the software. This paper describes how one performance evaluation system was developed by IBM Federal Systems Division to meet these needs. This system consists of a data base, a simulator, and an analytical model that also serves as a pre-processor to the simulator. The paper also describes the flexible features of each of these parts and provides illustrations of their use.[1]

1. INTRODUCTION

This paper describes a performance evaluation approach to quantify the performance of large, customized systems. The approach consists of a set of procedures and flexible tools used together to analyze performance throughout the system life cycle.

This general methodology evolved over the multi-year cycle of one complex, customized system. As development proceeded from preliminary design through coding and testing, a number of tools were developed and/or used to predict the expected performance of the operational system. The performance products ranged from pure estimation of system performance, to system measurement, and to modification analysis.

This paper is an expansion of an earlier paper [1], which reported an initial approach to estimation of stand alone timings. Because of the proprietary nature of the projects to which this performance effort was directed, a hypothetical example [Section 4. Example Development Effort (Library Automation Project)] has been included. Actual library data was used to calibrate the required model input data.

1.1. PERFORMANCE PROBLEM

This method addresses the problem faced by all large development efforts: the reduction of risk by the earliest possible identification of performance problems. Performance cannot easily be grafted onto a system in its final development stages, and any attempt to do so is expensive. Since early discovery of flaws can lower the cost of fixes by orders of magnitude, we needed to provide the earliest possible prediction, tracking, and analysis of system performance.

1.2. PERFORMANCE EVALUATION EFFORT

The performance effort described in this paper is divided into discussions of:

- The general performance approach to performance analysis of a large development project,
- Our specific implementation, and
- Our approach applied to a sample project.

2. DEVELOPMENT PROJECT FROM A PERFORMANCE VIEWPOINT

2.1. PROJECT LIFE CYCLE

The nature of performance analysis changes as a project proceeds through its life cycle with the requirement that the performance analyst play an integrated role throughout. Historically, the role of the performance analyst has usually been that of a late arrival resulting in limited impact on the project. The preferred role has performance directly involved with the development process from the outset. This permits performance to play a significant role in the direction of the development effort. Ferrari emphasizes this role in a recent IEEE article [2]:

> "... The performance evaluation viewpoint ... does not favor serendipity as the ordinary approach to (performance) problem solving, 'hacking,' or 'hack now, fix later' methodology."

2.1.1. PROJECT OUTSET (Design for Performance). The goals of the performance analyst at the project outset are:

- To identify critical performance parameters,
- To make initial estimates of these parameters, and
- To assist in the solution of problems identified by the initial estimates.

At the project outset, software architects will be assessing and modifying candidate designs at an amazing rate. The performance analyst must keep up with these furiously changing designs and, additionally, provide insight into their performance ramifications. To provide some method for this madness, the analyst must quickly focus on a method of quantifying performance and stick with it.

There are many candidate performance measures for system tracking (e.g., CPU utilization, transaction throughput, response time, system availability, required floor space, power). The trick is to identify an appropriate subset that will permit the practical tracking of project technical performance (current and future) through all system development phases.
One tracking methodology is presented in MIL-STD-499A [3]. This document describes how these Technical Performance Measures (TPMs) are selected, reported, and

---

[1]Permission is granted to CMG to publish this paper in its publications. IBM retains the right to publish this paper elsewhere or distribute copies of it to whomever it chooses.

utilized. The basic idea with TPMs is to reduce the performance tracking to a workable level (both from a resource and a management viewpoint) while accurately representing overall system performance. All of the performance values will be estimates at the project outset (but system improvements can be made at very low cost). The disadvantage of these early estimates is they rely on marginal input data. Some of the different inputs that must be estimated are:

- o Software characteristics (e.g., application path lengths),
- o Machine characteristics (e.g., internal throughput ratios),
- o Operator characteristics (e.g., time to perform various tasks at interactive screens), and
- o Environmental characteristics (e.g., impacts of the operating system or other applications).

2.1.2. PROJECT MIDCOURSE (Development with Performance Corrections). The goals of the midcourse effort are:

- o To advise management of performance problems as they are detected,
- o To keep the performance improvement process focused, and
- o To provide assistance for test plan generation.

After the initial designs are complete, the performance analyst must keep the performance effort focused as the code is written. Changes and problems will continue to surface, and, inevitably, management will not have sufficient resources to deal with all of them immediately. The analyst must identify the "Heavy Hitters" among the problems and advise management of the best application of its resources.

At a project's midcourse the measurement of actual code performance becomes possible. This measurement provides, at a high level, information on such items as task elapsed and CPU timings and at a lower level, individual subtask timings and I/O activity. The cost of making corrections to the code at this point is still considerably cheaper than after the code becomes operational. Caution in the use of the absolute value of measurement results is necessary at this midcourse because the applications and system software are untuned and will produce timing values orders of magnitude larger than the estimates initially generated and the expected results after system tuning.

2.1.3 PROJECT OPERATIONS (PERFORMANCE ENHANCEMENT). The performance goals of project operations are:

- o To understand and quantify how the system is actually used,
- o To maintain workload projections, and
- o To plan for system enhancement.

After the system becomes operational, the performance analyst is the focal point for analyzing changes to the system. The analyst must keep a current model of the system which can be used to study varying conditions and to plan for increasing workload.

When the system becomes operational the tuning will have been completed and the environment will have become realistic. Now, however, the cost of performance enhancements will have increased several orders of magnitude. The focus of performance will have shifted towards future growth through system upgrades and enhancements.

2.2 TOOLS NEEDED

The tools needed to quantify TPMs vary over the project development cycle. The tools fall into three categories:

- o Tracking,
- o Estimation, and
- o Measurement.

2.2.1 TRACKING TOOLS. A large application will always have many tasks and details associated with it. To keep track of these items for a performance effort, a data base must be employed. In the early design phase, the data base is used to keep track of the hierarchy of tasks being developed. As task data (requirements, model estimates, and measurements) becomes available, they can be stored in the data base as a central collection and communication point. Periodic and ad hoc reports can be prepared from the data base.

2.2.2. ESTIMATION TOOLS. The performance analyst needs some way of obtaining early TPM estimates. These estimation tools generally come in two flavors: analytical models and simulations. Analytical models, by expressing the operation of the system in mathematical terms, provide a quick picture of stand-alone performance. They can provide as much (or as little) detail as is required and can be tailored to the particular TPMs of interest. Simulations permit a more customized representation of a system, but require more effort. The two tool types complement one another.

2.2.3. MEASUREMENT TOOLS. Measurement of task performance first becomes feasible during early testing, and can be performed in a number of ways. The operating system will provide standard accounting capabilities that can be used to calculate some performance parameters. In addition, specialized performance software is available from a number of vendors to augment the data supplied by the accounting tools. Both sets of tools are useful when applied together, but the data must be correlated. In addition, measurement data can be used to verify the estimation results and to improve the accuracy of the inputs to the estimation tools.

3. OUR SPECIFIC APPROACH

3.1 OUR PROJECT LIFE CYCLE

Our project covered several years during its multi-phase development. We worked in parallel with the project developers through two groups. The first, the Performance Working Group, was formed to explore the technical issues behind performance problems. The second, the Heavy Hitters Group, was formed to provide a management focus to the performance work.
In the early design phases estimates of run duration were provided and bottlenecks were identified. As the project moved to coding and testing, this parallel work continued and expanded to include collection of measurements. Care was taken to gather data as "targets-of-opportunity" became available to minimize additional test workload. As measurements were gathered during functional testing, results were analyzed to determine whether performance was within appropriate bounds.

When early measurements indicated potential performance problems, the code was examined for "quick-fix" problems (e.g., problems with parameter settings or data base access choices which could be corrected without changing

source code). Any problems of this category were fixed and functional testing continued.

If these measurements indicated that a task still would not meet operational performance requirements, then the task code was moved to a performance testbed and analyzed further. There the Performance Working Group ran instrumentation to isolate performance problems and assist in evaluation of suggested solutions.

3.2. OUR TOOLS

The specific tools chosen for our project were a collection of standard commercial packages and locally produced code.

3.2.1. TRACKING TOOLS. The performance data base was developed in PC FILE III [5] to provide for the easy retention of the collected task timeline information. Task characteristics included parameters such as I/O activity volume and type (e.g., random fetch, sequential update); CPU cycles at task start, midcourse, and task end; and test measurements. These characteristics were upgraded from rough estimates during preliminary design phase to tuned measurements as the code moved through the development process.

3.2.2. ESTIMATION TOOLS.

3.2.2.1. Analytical Summary Model. Four user defined parts compose the analytical summary model:

- o A set of variables describing software, hardware, operator, and environmental characteristics;
- o A set of equations describing resource requirements for each defined subtask;
- o A list of all other subtasks called by a given subtask and the quantity of these calls; and
- o A recursive algorithm that combines the variables and the equations to represent the structure of the actual code.

Further discussion of this analytical structure can be found in [1]. The outputs of these models were retained in the performance data base.

3.2.2.2. Discrete Simulation Contention Model. A discrete simulation model was developed to study system operation under a variety of timeline scenarios. This simulator, coded in PL/I [8] with imbedded GPSS macros [9], represents the baseline configuration illustrated in Figure 1. It includes a CPU, strings of Direct Access Storage Devices (DASD), controllers, and operators. These resources appear generically in the simulator code. A set of input files is superimposed on this baseline to define the specific operational system being simulated.

3.2.2.2.1. Inputs. Three types of input files are used by the simulator: a system characteristics definition, an applications software description, and a scenario description. The system characteristics definition includes hardware specifications (such as processing power and average DASD access rate) and system software specifications (such as operating system overhead). In this description the task may be treated as a single entity (e.g., CPU processing) or it may be decomposed into hundreds of steps with a description of the resources needed for each.. The scenario description lists application software tasks run during the simulation and the start time of each. The application software description lists the resources needed by each task. The analytical model produces this description as one of its outputs.

3.2.2.2.2. Processing. The simulation permits multiple strings of tasks to contend with one another for limited system resources.

3.2.2.2.3. Outputs. The model provides its results in two forms: the standard GPSS reports and the timeline reports generated in the PL/I code. The standard reports give a variety of statistics on resource use including number of service requests per resource, average service time per request, and average queuing time. The timeline report lists the start, end, and duration of each task.

3.3. OUR MEASUREMENT TOOLS

Standard monitors such as RMFMON [6] and OMEGAMON [7, Note: OMEGAMON is a trademark of the Candle Corporation] are used to obtain such measures as task CPU, I/O, and elapsed times. These measurements are retained in the performance data base. These results provide insight into potential performance problems and performance tuning progress.

4. EXAMPLE DEVELOPMENT EFFORT (LIBRARY AUTOMATION PROJECT)

4.1. SYSTEM OVERVIEW

We have chosen a library automation package to illustrate how these tools are used for software development. The example library system handles a variety of tracking and inventory functions for a regional library in a large county system. Per region the system has approximately 100,000 patrons and 200,000 volumes. A uniprocessor with DASD, library work stations with bar code wands, and conventional terminals was selected to service these functions (Figure 1).

4.2. FUNCTION TO BE ANALYZED

The timeline of interest for this example is the crux of the system: the patron's service function. This timeline is illustrated in Figure 2 - Technical Performance Measures (TPM) Budgets. The figure shows the types of data kept in the timeline data base. Each portion of the timeline is given a specified budget and an average data base size for which the budget applies. For this patron service procedure the timeline is broken into three sequential tasks: a patron status check, a checkout/return activity, and a statistical data update. For this example we will focus on the evolution of the checkout/return task. This task was deemed most critical and most difficult to perform within the required time constraints. A high-level flow design of this task is given in Figure 3 - Checkout/Return Task Flow Chart. The librarian initiates the checkout/return processing and proceeds through the task as described below.

4.2.1. Checkout/Return Initialization. After patron status check is complete, checkout/return initialization is invoked and information on overdue books, unpaid fines, and total books checked out is passed to the task.

Initialization includes locking the patron's record and establishing a checkout/return workspace.

4.2.2. Borrow Sequence. If the librarian selects the borrow sequence, he/she next reads in a book identification code. If the book is a reserve, the patron's request in the reserve list is cleared. The patron and the volume files are then updated as appropriate. The librarian may loop back through this procedure. When this sequence is complete, the statistical data update task is invoked to complete the patron's service function.

4.2.3. Return Sequence. If the librarian selects the return sequence, he/she reads in the book identification code, and the patron and volume files are updated. If the book is on the reserve list, the librarian will receive a message to hold the book. If the book is overdue, the fine is calculated and displayed. The librarian may also loop back through this procedure, and the total fine (if any) will be accumulated. When this sequence is complete, the statistical data update task is invoked to complete the patron's service function.

4.3. APPROACH APPLICATION

As the code proceeds through its life cycle the models are exercised in several ways to assist in project development and maintenance. Three examples of its utility are given.

4.3.1. Early Design Phase. For modeling purposes the code was described as in Section 3.2. The initial versions of these files are given in Figures 4 through 6. The working constants are listed in Figure 4. Figure 4 exhibits the values assigned to the variables, used in the procedure definition in Figure 5 (Checkout/Return Task Definition) and the call sequence in Figure 6 (Checkout/Return Call Sequence). Running the model with these inputs produced the report given in Figure 7 (Model Output).

This initial report highlighted a problem in both the borrow and return checks. A bottleneck at disk 4 caused the elapsed task time (147.6 minutes) to exceed its 9 second budget. This problem prompted the performance working group to analyze the design further and uncovered the reason for the problem, the implementation of the reserved list data base. The reservation information had been added to the patron file, requiring that on the average, half of the patron file be checked before a book's reservation status could be determined. The design was updated by moving this information to the volume data base and adding a pointer to patron information. The affected task definition equations were updated (Figure 8), and the model was run again reducing the run time to under 8 seconds.
Studying the design under contention next, all five operators were simulated simultaneously. With the baseline processor speed of 2.4 MIPs produced Contention Run 3.A. A second processor with a speed of 3.6 MIPs produced Contention Run 3.B. These runs show the additional compute speed did not accomplish much because the operators were the limiting resource.

4.3.2. Midcourse Development Use. As the code was generated initial measurements were made by the testers. These measurements exhibited values several times larger than the required response times. This is an expected result given the untuned state of the system. The measurement data provided model input which was used to generate a first curve (Figure 9, Line #1).

The performance fixes at this point are typically easy to locate and not too expensive to fix in terms of time and code change (e.g. data file placement, data transfer blocking factors, search algorithms). Some of these changes were then placed in the computer model and predicted improvements were obtained, Figure 9, Line #2. The results of these quick fixes were implemented in the prototype subsystem. Retesting indicated that the predictions were correct.

At this stage the results still did not exhibit either in the model or in the "piggybacked" tests the desired 9 second response over a sufficient range of terminals. A task force composed of performance working group members and system development personnel was established. The task force conducted controlled tests and analyses to identify where the remaining performance problems lay. Additional tuning improvements to the task were implemented in the system. Measurements were taken and these result appear in Figure 9, Line #3.

Testbed measurements were placed in the model and modified to reflect final tuning of the system. The model results indicated that the system should be able to meet the stated budgets at system installation. These results are exhibited as Figure 9, Line #4.

4.3.3. Operational Performance Use. In the development phase, model prediction was used to quantify performance. As the development cycle progressed, emphasis shifted to measurement, and by completion of the development essentially all results were based on measurement. The simulation capability, however, continued to play a direct role in predicting the need for future software and hardware enhancements.

The contention modeling capabilities of the simulator were used to evaluate the processor configuration required to meet an increased workload demand. A specific configuration with a specified processing capability, memory size and DASD size was selected. The model was used to determine whether the proposed configuration could handle the upgraded load of seven terminals. The checkout/return task was run in the simulation with 7 operators handling a peak period influx of patrons. The simulated elapsed time for the checkout/return task was 8.3 seconds. The CPU was 15% busy, a low value; however, the channel loading of 18% indicated a probable system maximum of 10 operators performing the checkout/return task.

5. APPROACH ADVANTAGES SUMMARY

Our experience has shown the approach provided valuable project support. It accurately identified problem areas and directly assisted in their resolution with minimal schedule impact.
This approach includes a number of features which make life easier for the performance analyst. It has the flexibility to work with whatever information is available. Tasks can be modeled in great or limited detail depending on what is known or what level of detail suffices to answer a given question. Tasks can be modeled singly to determine isolated run times or in groups to simulate contention for limited system resources. The modeling tools remain useful over the entire life cycle. As measurement data becomes available or requirements change, they can be folded into existing models. These features significantly increase the effectiveness of the performance analyst throughout the system life cycle.

There is, however, a threshold of project size and/ or complexity to cross before this approach becomes feasible. Any size project would benefit from the application of parts of the approach (within budget constraints). Because of the start-up costs of implementation, the application of the complete approach requires the financial coverage of a large project (producing several hundred thousand lines of code and spanning several years of development).

REFERENCES

[1] W. Godwin and W. Suhler, "A Timing Estimation Method for Large System Software Development," Computer Measurement Group (CMG), December, 1985 (Proceedings).

[2] D. Ferrari, "Considerations on the Insularity of Performance Evaluation," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol SE-12, No. 6, June 1986.

[3] DOD Military Standard 499A(USAF) Engineering Standard, 17 July 1969.

[4] USAF DI-3619/S153, Technical Performance Measurement Report, 9 February 1970.

[5] PC-File III Data Base Program, Button Ware Inc.

[6] OMEGAMON, Candle Corp, Los Angeles, California.

[7] RMFMON- IBM Program Product.

[8] PL/I- IBM Program Product, OS PL/I Optimizing Compiler

[9] GPSS PL/I- IBM Program Product.

FIGURE 1 – SYSTEM CONFIGURATION

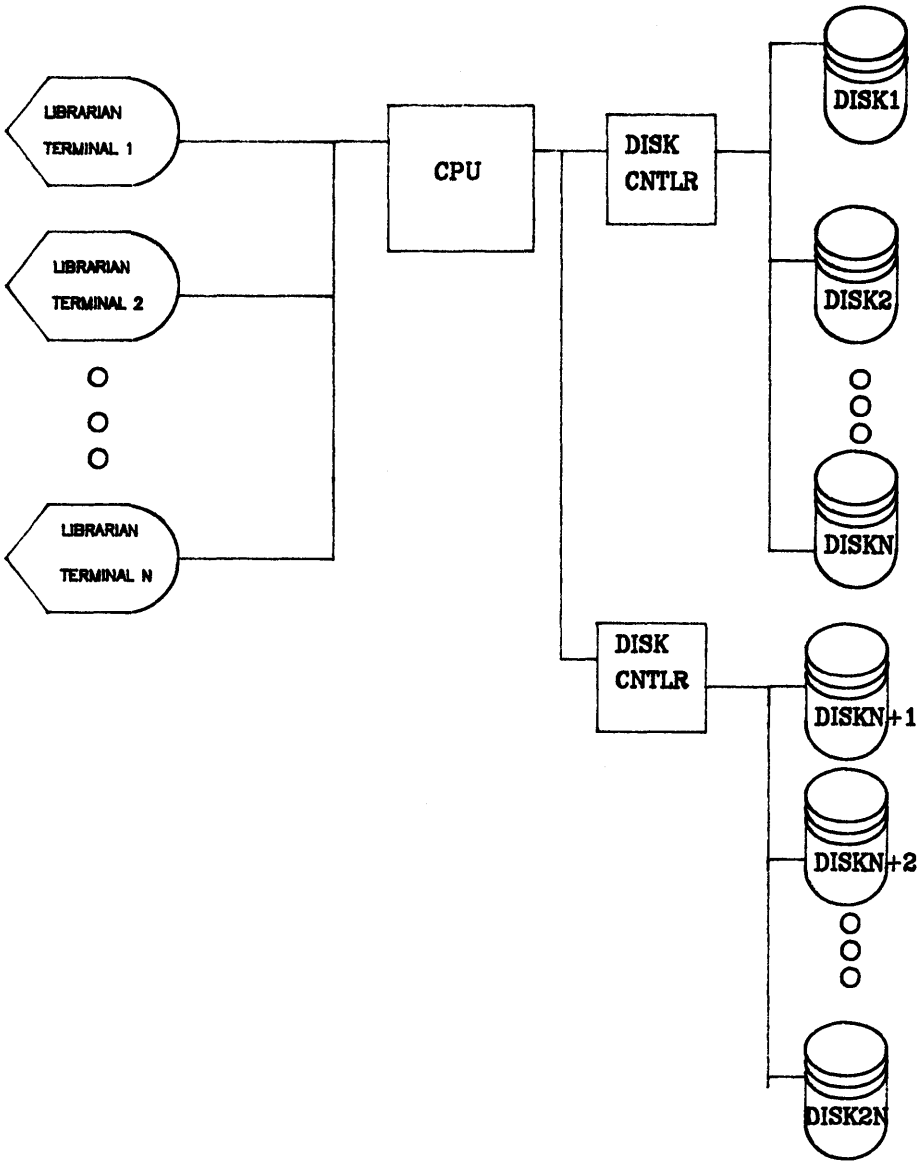
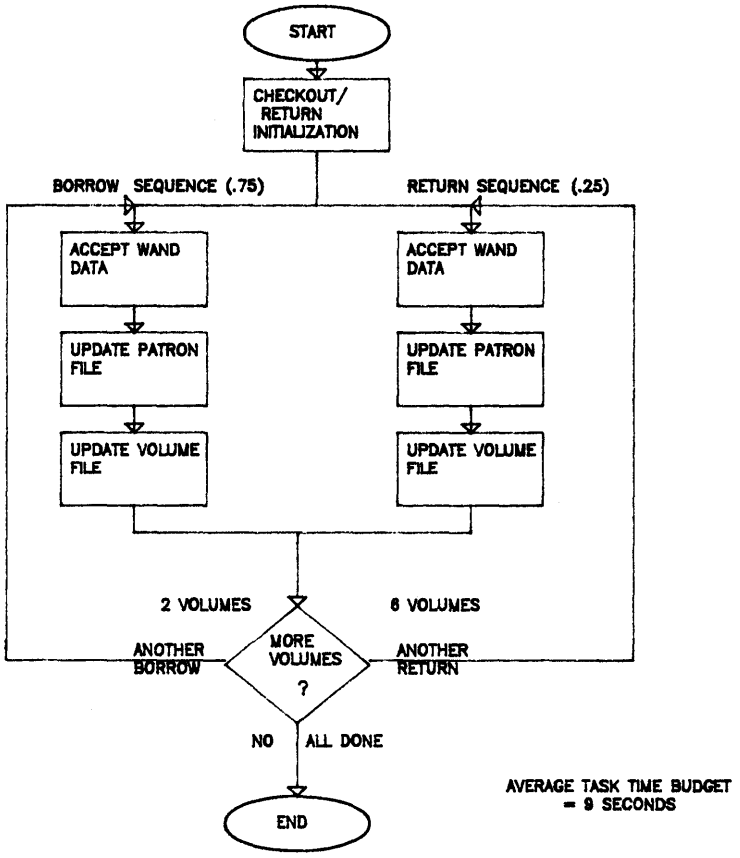


FIGURE 2- TECHNICAL PERFORMANCE MEASURES (TPM) BUDGETS

TPM #	TPM Type	BUDGET	DATA BASE SIZE
1	Patron status check	5 seconds	100 K patrons per region
2	Checkout/return task	3 seconds per volume	200 K volumes per region
3	Statistical data update	2 seconds	200 K volumes per region

FIGURE 3– CHECKOUT/RETURN TASK FLOWCHART

(FROM PATRON STATUS CHECK TASK)



(TO STATISTICAL DATA UPDATE TASK)

FIGURE 4 - Working Constants

```
TOTAL_PATRONS = 100000        'Number of patrons in system
TOTAL_VOLUMES = 200000        'Number of volumes in system
PROB_BORROW   = 0.75          'Probability of borrowing
VOLUMES_OUT   = 2             'Average number of volumes borrowed per visit
PROB_RETURN   = 0.25          'Probability of returning
VOLUMES_IN    = 6             'Average number of volumes returned per visit
MIPS          = 2.4           'Millions of instructions per second
DASD_TIME     = 28            'Milliseconds
OVERHEAD      = 0.20          'Operating system overhead
```

FIGURE 5 - Task Definition

```
'*****************************************************
'
'            CHECKOUT/RETURN TASK DEFINITION
'
'*****************************************************
K = 1
CHKRTRN = K              'Checkout/Return Resource
                         'Definition
PATH(K) = 5              'Path Length,1000's of
                         'Instructions
OPTIME(K) = 5            '5 Second Operator Time

K = K + 1
BORROW = K               'Borrow Sequence Definition
PATH(K) = 5              'Borrow Path Length
FRACC(K) = 2             'Random Fetch Accesses

K = K + 1
RETURN = K               'Return Sequence Definition
PATH(K) = 5              'Return Path Length
FRACC(K) = 2             'Random Fetch Accesses

K = K + 1
WAND_DATA = K            'Accept Wand Data Definition
PATH(K) = 10             'Wand Data Path Length
FRACC(K) = 1             'Random Fetch Accesses

K = K + 1
PATRON_BORROW = K        'Update Patron Borrow
PATH(K) = 10             'Patron Borrow Path Length
FSACC(K) = (0.5*TOTAL_PATRONS)  'Sequential Fetch
                         'Accesses
IRACC(K) = 1             'Random Insert Accesses

K = K + 1
PATRON_RETURN = K        'Update Patron Return
PATH(K) = 10             'Patron Return Path Length
FSACC(K) = (0.5*TOTAL_PATRONS)  'Sequential Fetch
                         'Accesses
DRACC(K) = 1             'Random Delete Accesses

K = K + 1
VOLUME_BORROW = K        'Update Volume Borrow
PATH(K) = 10             'Volume Borrow Path Length
URACC(K) = 1             'Random Update Accesses

K = K + 1
VOLUME_RETURN = K        'Update Volume Return
PATH(K) = 10             'Volume Return Path Length
URACC(K) = 1             'Random Update Accesses
```

FIGURE 6 - Checkout/Return Call Sequence

```
'*****************************************************
'
'            CHECKOUT/RETURN CALL SEQUENCE
'
'*****************************************************


K = CHKRTRN

SCALL(K,1,0) = BORROW            'Call to Borrow Module
SCALL(K,1,1) = PROB_BORROW       'Number of Calls
                                 '(Probability)

SCALL(K,2,0) = RETURN            'Call to Return Module
SCALL(K,2,1) = PROB_RETURN       'Number of Calls
                                 '(Probability)
 K = BORROW
SCALL(K,1,0) = WAND_DATA         'Call to Wand Data
                                 'Module
SCALL(K,1,1) = VOLUMES_OUT       'Average Number Checked
                                 'Out

SCALL(K,2,0) = PATRON_BORROW     'Call to Patron Borrow
                                 'Module
SCALL(K,2,1) = VOLUMES_OUT       'Average Number Checked
                                 'Out

SCALL(K,3,0) = VOLUME_BORROW     'Call to Volume Borrow
                                 'Module
SCALL(K,3,1) = VOLUMES_OUT       'Average Number Checked
                                 'Out

K = RETURN

SCALL(K,1,0) = WAND_DATA         'Call to Wand Data
                                 'Module
SCALL(K,1,1) = VOLUMES_IN        'Average Number
                                 'Returned

SCALL(K,2,0) = PATRON_RETURN     'Call to Patron Return
                                 'Module
SCALL(K,2,1) = VOLUMES_IN        'Average Number
                                 'Returned

SCALL(K,3,0) = VOLUME_RETURN     'Call to Volume Return
                                 'Module
SCALL(K,3,1) = VOLUMES_IN        'Average Number
                                 'Returned
```

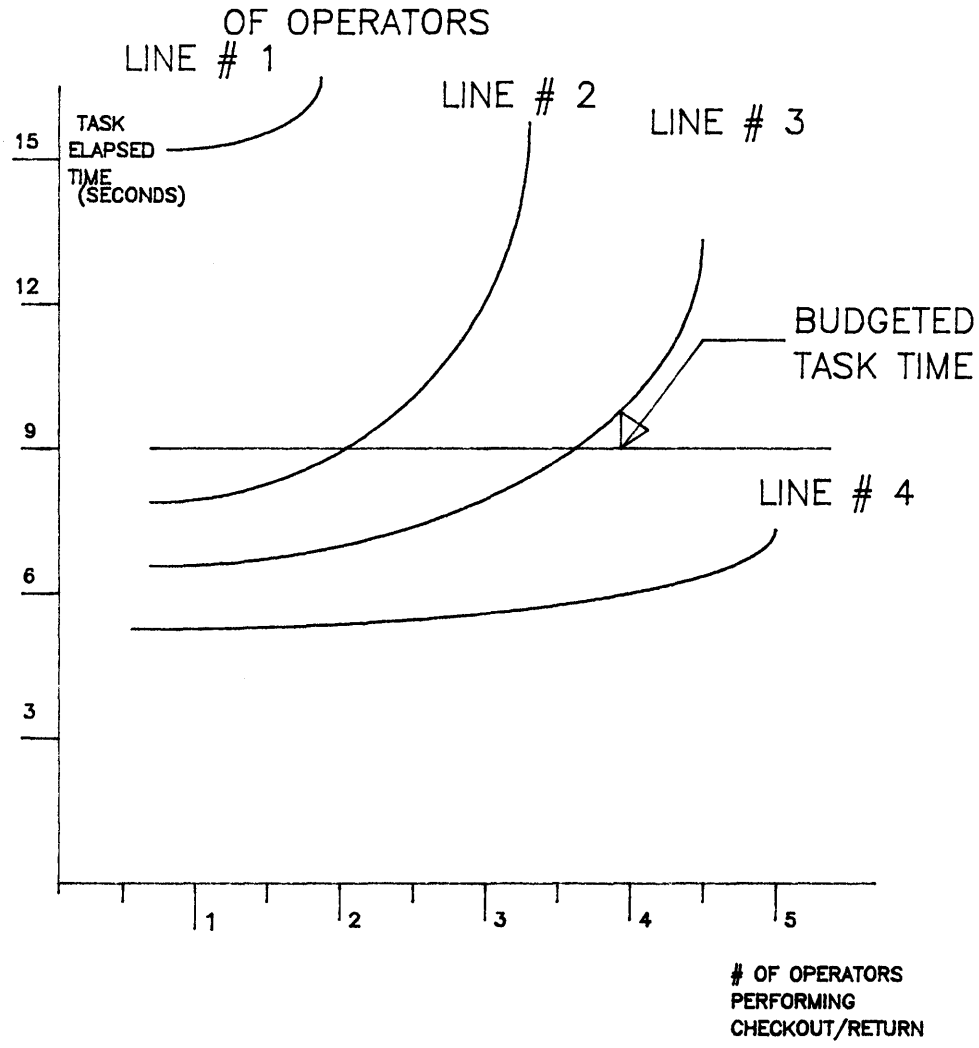
FIGURE 7 - Model Output

RUN	RUN 1 INITIAL DESIGN	RUN 2 MODIFIED DESIGN	RUN 3.A CONTENTION (5 Oprtrs) 2.4 MIPS	RUN 3.B CONTENTION (5 Oprtrs) 3.6 MIPS
CPU	6.8%	3.0%	15.2%	11.7%
CHANNEL1	13.5%	4.5%	18.8%	19.3%
CHANNEL2	0.0%	0.0%	2.1%	2.0%
DISK 0	4.7%	10.5%	35.9%	38.0%
DISK 1	0.0%	0.0%	18.1%	18.9%
DISK 2	0.0%	0.0%	9.9%	9.5%
DISK 3	0.0%	5.0%	8.6%	9.1%
DISK 4	80.4%	3.8%	37.6%	36.5%
DISK 5	4.7%	6.0%	18.4%	19.2%
DISK 6	4.7%	6.0%	9.6%	9.1%
DISK 7	0.0%	4.5%	8.7%	8.1%
OPERTR 1	0.0%	60.8%	59.6%	60.6%
OPERTR 2	--	--	58.9%	59.9%
OPERTR 3	--	--	58.3%	59.3%
OPERTR 4	--	--	57.2%	57.7%
OPERTR 5	--	--	56.2%	56.7%
TOTAL RUN TIME (MINUTES)	147.6	0.1	15.0	15.0

FIGURE 8 - Modified Task Definition

```
'*****************************************************
'
'       MODIFIED CHECKOUT/RETURN TASK DEFINITION
'
'*****************************************************

K = 1
CHKRTRN = K        'Checkout/Return Resource Definition
PATH(K) = 5        'Path Length,1000's of Instructions
OPTIME(K) = 5      '5 Second Operator Time

K = K + 1
BORROW = K         'Borrow Sequence Definition
PATH(K) = 5        'Borrow Path Length
FRACC(K) = 2       'Random Fetch Accesses

K = K + 1
RETURN = K         'Return Sequence Definition
PATH(K) = 5        'Return Path Length
FRACC(K) = 2       'Random Fetch Accesses

K = K + 1
WAND_DATA = K      'Accept Wand Data Definition
PATH(K) = 10       'Wand Data Path Length
FRACC(K) = 1       'Random Fetch Accesses

K = K + 1
PATRON_BORROW = K  'Update Patron Borrow
PATH(K) = 10       'Patron Borrow Path Length
IRACC(K) = 1       'Random Insert Accesses

K = K + 1
PATRON_RETURN = K  'Update Patron Return
PATH(K) = 10       'Patron Return Path Length
DRACC(K) = 1       'Random Delete Accesses

K = K + 1
VOLUME_BORROW = K  'Update Volume Borrow
PATH(K) = 10       'Volume Borrow Path Length
URACC(K) = 1       'Random Update Accesses

K = K + 1
VOLUME_RETURN = K  'Update Volume Return
PATH(K) = 10       'Volume Return Path Length
URACC(K) = 1       'Random Update Accesses
```

FIGURE 9– ELAPSED TIME VERSUS NUMBER OF OPERATORS



SIZING NEW AUTOMATED SYSTEMS HAVING UNKNOWN WORKLOADS: A STRUCTURED APPROACH

John L. Gibson
Federal Computer Performance Evaluation and Simulation Center

ABSTRACT

At some point, every business identifies new applications requiring automation. Two critical issues affect the costs of automation: defining the workload to be automated and estimating the required computer resources.

This paper presents a structured approach for projecting the hardware requirements to support new applications. Four steps were used in this approach: (1) workload definition in Natural Business Units (NBU's); (2) NBU translation to data processing workload units; (3) workload volume estimation; and (4) resource requirements estimation. This approach is practical for business environments of any size and with varying automation requirements.

1. INTRODUCTION

Capacity management is becoming a means of existence for Federal government data centers these days. The objective is classic: ensuring data centers provide adequate service to users today, tomorrow, next year, ad infinitum. Yet, long hardware and software procurement cycles, budget constraints, administration fluctuations, political maneuvering: all of these issues and more contribute to Federal data processing capacity illnesses that are only treated and never cured. Life for the capacity planner is pretty frustrating!

Occasionally, though, a slightly different capacity question appears to spur one's creativity and interest. This paper describes one such question: sizing systems for unknown workloads. The Federal Computer Performance Evaluation and Simulation Center (FEDSIM) was recently asked by a major government agency to address this issue.

Sizing systems for unknown workloads may seem to pattern the proverbial "chicken before the egg" scenario. Therefore, some additional background is presented below in three parts: (1) a statement of the problem; (2) a preliminary strategy; and (3) some of the risks identified early in the project.

1.1 THE PROBLEM

Suppose you have been given the responsibility for automating a manual process that represents a fundamental portion of your business. During an initial presentation to corporate management, you focus on the potential advances in productivity, enhancements to information accessibility and timeliness, and improvements to business scheduling and process control--all resulting from your automation plan. Suddenly, from the back comes a simple, but fair, observation: "I like the concept, but what are the hardware requirements?" Since your objective at this point was only to sell the automation plan, you may not have a definitive answer. But, your management expects the results from both sides of a cost/benefit analysis. Your next task has been clearly identified.

Determining the size of new automated systems is a complex problem that presents new challenges to the capacity planner. Typically, the capacity planner is concerned with future resource requirements based on workload forecasting and trending. Historical data are the basis for analyses and projections. For new systems, however, no historical perspective exists. The application design, the user community, and the service requirements may all be unknown or not initially defined. Is an estimate of system size possible?

1.2 A STRATEGY

A large number of unknowns tends to cause procrastination for many capacity planners. From their view, the cost of analysis exceeds the anticipated accuracy of the results. However, one must remember the sizing objectives at this point: sizing estimates will be used by management to consider the feasibility of automation. In other words, a sizing analysis will estimate the "class" of machine (thus, a hardware cost range) necessary for the new automated process. Further, the primary management decision resulting from such a study might simply be to continue or abandon additional analysis plans. The key idea here is to discount neither the quality of analysis nor the importance of the results due to the unknowns. On the contrary, capacity planners must quantify the importance of the decisions resulting from their projections and adjust the depth of their analysis strategies accordingly.

Now that you're convinced a sizing analysis for unknown workloads is worthwhile, let's discuss possible objectives and strategy. Typical objectives might be: (1) estimating workload volumes and resource requirements that the new application will generate when operational; and, (2) estimating the "class" of machine required to process the new workload with an assumed service range.

To address these objectives, FEDSIM developed a sizing strategy that included these four parts: (1) defining the workload; (2) translating the workload to data processing terms; (3) estimating the workload volume; and (4) estimating the resource requirements. A detailed discussion of this strategy is in Section 2 below.

1.3 THE RISKS

Two major risks were inherent in the strategy outlined above. First, data from the manual systems for projecting workload volumes may be nonexistent or of poor quality; these data (if available) may be inapplicable to the automated system due to procedural differences. Second, if the functional design of the automated system is unknown and no implementation plans have been made, it may be impossible to identify suitably equivalent application environments for obtaining resource usage estimates.

These risks are highlighted to indicate the speculative nature of the sizing strategy. However, the identified risks should be weighed against the importance of the resulting decisions before prejudging the validity of the analysis.

2. SIZING METHODOLOGY

This section outlines the approach used for identifying, analyzing, and sizing the data processing requirements for currently manual workloads. This methodology requires four tasks: (1) identifying the major procedures, processes, and/or events that comprise the manual workload; (2) translating the manual activities to data processing work units (e.g., transactions); (3) estimating the volume and distribution of the transactions during normal system operation; and (4) characterizing the transactions by data processing resource requirements (e.g., CPU, I/O, on-line storage). Each of these tasks is described below.

2.1 WORKLOAD DEFINITION

Sizing new workloads is impossible without thoroughly researching the manual processes planned for automation. The objectives of this research should include: (1) understanding the functional work components comprising the manual activities; (2) identifying the existing interfaces among the manual processes and currently automated systems; and (3) establishing the necessary background for understanding the functional requirements of the new automated system. This task consists of two stages: data collection and data minimization.

2.1.1 DATA COLLECTION. A complete review of documentation, standards, and procedures is mandatory. If the new automated system is already under development, a review of the functional design specifications prepared thus far should address much of the background information.

During the background research, attention should be given to reviewing any and all management workload reports describing the manual system. These reports should be gleaned of all quantitative measures such as volumes and trends that may later become the basis for estimating the automated workload volume and resource requirements.

Personnel responsible for the automated system design should be interviewed to discuss any design features that might affect workload volume. Furthermore, the end users should be interviewed to discuss their expectations from the automated system, such as increased productivity and improved management/control. For the capacity planner, this task should result in a functional workload profile of the planned automated system. This profile will identify the most fundamental characteristics of the manual workload (e.g., source, rate) and will define the generic workload requirements resulting from automation (e.g., processing type [batch or interactive], application type [database, process management]).

2.1.2 DATA MINIMIZATION. During the data collection phase, the capacity planner may discover that a large number of variables associated with the manual processes appear to potentially influence the automated system. A significant amount of time can be spent analyzing and debating the value of each variable. Once again, the amount of time available for analysis and the criticality of the result must be weighed against the required accuracy. If the new automated system is still in early stages of design, I suggest that the greater return from the sizing analysis comes not from the absolute quantification of the workload, but rather from identifying which variables have greater impact on workload volumes. What's needed is a tool to assist in quickly eliminating the unrelated variables and reducing what's left to a manageable number. The analysis of variables using linear regression is one approach.

Regression analysis is a technique for discovering empirical relationships between a dependent variable (e.g., workload) and one or more independent variables or factors (e.g., manual processes or sub-processes). Regression analysis finds the equation that best fits the available data and minimizes residual errors due to measurement error or factors not included in the model. Further, using the stepwise regression method systematically adds or removes factors from a regression model depending on whether the factors make a statistically significant contribution to the model's accuracy. Through this mathematical technique, the major manual processes influencing the anticipated automated workload can be identified to minimize the translation process from the manual functions to the automated workload.

2.2 WORKLOAD TRANSLATION

The workload translation from manual functions to data processing transactions involves two steps. First, each of the major activities currently performed manually is related to its automated counterpart. Second, the functional design requirements or known system components are analyzed to determine the definition(s) of automated work units in the automated environment.

2.2.1 MANUAL WORKLOAD MAPPING. Meetings between the capacity planner, system designers, and end users must be held to determine how each major manual workload might be supported and/or performed by the automated system. A synergistic approach can be used to develop workload scenarios from a user perspective for each manual function. Then, using a worksheet approach, a matrix can be produced to define these relationships: (1) what automated activities are required for each manual function; (2) the number and type of automated work units (e.g., screen displays, processes) resulting from the execution of each manual function; and (3) the anticipated user input requirements and output expectations. Comparisons of these scenarios allow the analysts to visualize how the automated system will support its users and to identify repetitious and/or overlapping workloads.

2.2.2 AUTOMATED WORKLOAD DEFINITION. Following the manual workload analysis, the capacity planner probably will rely on his/her data processing experience in comparing the expected user requirements from the new system with other existing automated systems. The objectives of this comparative study should be to determine the initial workload definitions for the new automated system in terms of functional capabilities. For example, this analysis should generally determine the type of automated system required (e.g., batch, interactive), the data sources and data management requirements (e.g., data base architecture), and the categories of anticipated functional work units (e.g., transaction types). These determinations provide the basis for defining the automated system workload in some quantitative measures and distributing the workload into categories according to estimated data processing requirements.

2.3 WORKLOAD VOLUME ESTIMATION

Initial workload volume estimates for the new automated system are possible using the results of the two components of the workload translation process

described above. Management reports and/or documentation on the manual functions must be scrutinized for work volume indicators. The relational matrix produced during the manual workload mapping process provides the vehicle for estimating automated system transaction volume and distribution.

Now is also the appropriate time during the workload analysis to discuss workload growth projections. The manual function documentation may provide some historical perspective to workload growth. However, one must understand that growth projections for the automated system may not be a direct extrapolation from the manual system growth statistics. For example, consider the latent demand issue. Manual system production delays may be occurring due to staffing and/or scheduling constraints. The automated system may improve operational efficiency to the extent that workload queues diminish. Thus, the potential for additional system throughput implies both a higher automated system transaction volume and a variable distribution of transaction types.

2.4 RESOURCE REQUIREMENTS ESTIMATION

Estimating accurate system resource requirements for the automated workload is extremely difficult. Many variables can affect these estimates, including database design, application design, database management discipline, and communications design. Often, a new system design does not dictate a specific hardware and software solution. Therefore, any number of system architectures could be proposed by competitive bid. In such cases, resource estimates are possible only by ad hoc selection of a reasonable system architecture (i.e., database system, user software, etc.) to support an application bearing resemblance to the new automated system. Using this system design and the derived workload definitions, private commercial companies and government agencies operating similar applications can be contacted with specific inquiries regarding the typical resource usage for transactions matching the workload definitions.

3. A CASE STUDY: ANALYSIS AND FINDINGS

This section provides a recent case study conducted by FEDSIM that demonstrates the sizing methodology and clarifies many of the analytical subtleties.

3.1 ENVIRONMENT DESCRIPTION

A major government agency is responsible for the financial management and administration of a huge cash reserve account. This account is under continuous audit to ensure proper cash flow, very similar to the practices incorporated by banks to manage short-term loan reserves. A portion of the account management involves follow-up's on delinquent client payments and client financial background investigations. This portion of the account management workload is a manual, paper intensive process. The agency is currently unable to adequately manage all active receivables to the account due to staffing shortages and the magnitude of the manual workload. Thus, the current operations concentrate only on the high dollar, high risk receivables.

Agency management viewed this practice as unacceptable to their mission responsibilities and elected to automate the manual portion of the account management system. Logically, the issues following this decision included an estimate of the automated system costs, which led to a workload sizing study.

3.2 NEW APPLICATION DEFINITION AND DESIGN

FEDSIM initially reviewed the system design plans and held in-depth discussions with agency personnel to determine a general definition and scope for the new automated system (hereafter, called the "XYZ system"). This information indicated that the XYZ functional requirements had a close resemblance to the large database management systems typically found in data processing today. It was also determined that the manual system workload was distributed among 20 geographically disjoint regions around the United States. The automation plans included analyzing and estimating the host system capacity and the on-line database storage requirements at each of the 20 sites.

FEDSIM also determined from the initial information gathering that the operational environment for the XYZ system would likely be interactive information retrieval, display, and update. The functional requirements of the system would include the management of account receivables, provide a repository for historical case information, assist in the scheduling and priority of investigative work, and highlight accounts impacted by pending Federal legislation.

3.3 SIZING RESULTS

Sizing the XYZ workload for each of the 20 host systems involved four steps: (1) defining the XYZ workload; (2) translating the XYZ definitions to data processing workload; (3) estimating the total transaction volume and the distribution of transaction types for each XYZ location; and (4) estimating the on-line storage requirements (i.e., database size) for each XYZ location. Each step is described below.

3.3.1 WHAT'S THE WORKLOAD. FEDSIM used the agency's description of its account management activities to guide the development of a set of independent variables for estimating the dependent variable, XYZ workload. FEDSIM sought to obtain data representing as many of the major XYZ modules being developed as possible. Data were extracted from the account activity report by geographic region for each of the years 1983 through 1986. Statistical analysis and historical workload comparisons indicated that these data were a reasonable representation of the baseline XYZ workload.

From the documentation and discussions with agency personnel, FEDSIM found that two major activities comprised the XYZ workload: accounts receivable processing and accounts investigation processing. The majority of the available data to theoretically explain the XYZ workload was included in an account activity report prepared for agency management. Three major statistics were found to quantify the accounts receivable work category. These figures were the total active and queue receipts, total active and queue dispositions, and total active and queue inventory. The total active and queue receipts was an appropriate accounts receivable determinant since it represented the volume of potential work arriving on the XYZ system. Components of the accounts receivable portion of the XYZ workload used the following nomenclature in the account activity report: full payments, installment agreements, staff hours, and uncollectible totals. The account activity report also identified quantifiable statistics for the investigations activity. These statistics were staff hours and total dispositions.

As described above, FEDSIM selected as many variables as possible from the account activity report that might influence the XYZ workload. Next, using a stepwise regression technique (i.e., PROC STEPWISE)

from the Statistical Analysis System (SAS), many of these variables were later rejected since they did not significantly affect the dependent variable. A linear regression model using PROC GLM was then developed to determine the workload relationships (i.e., coefficients for the independent variables). Two objectives were met using this model. First, the XYZ workload could be explained through a reduced set of tasks related to the XYZ software modules under development. Second, the estimated model coefficients enabled FEDSIM to derive transactions totals and storage requirements for each XYZ location.

The linear model to explain the XYZ workload eventually took the following form:

WKLD = UC + INS1 + INS2 + PENALTY + INV + OVERHEAD,

where,

WKLD	=	yearly accounts receivables + investigations per geographic region;
UC	=	yearly uncollectible totals per region;
INS1	=	yearly small account installment agreements per region;
INS2	=	yearly large account installment agreements per region;
PENALTY	=	yearly penalty accounts per region;
INV	=	yearly investigations per region;
OVERHEAD	=	Other processing over and above the five major tasks required by XYZ and system overhead required to run XYZ.

The statistics for these variables were all taken from the account activity report.

Further refinements to the SAS model using PROC GLM produced the following linear relationships among the contributing variables:

WKLD = .0058 * UC + 3.66 * INS1 + 5.78 * INS2 + 3.81 * PENALTY + 1.92 * INV + 1846

Model F = 560.32
R-Square = 0.96

The variables INS1, INS2, PENALTY, and INV were all found to be significant at the 99 percent level. The Model F indicates that the entire model as constructed was significant at the 99 percent level. Finally, the R-Square indicates that 96 percent of the variation in the dependent variable was explained by the estimated model. This group of statistics indicates that the XYZ workload was well explained using a minimal number of variables in the model. The independent variables were used to project XYZ transactions, given that workload patterns and the accounts receivable operations procedures do not change.

3.3.2 WHAT'S THE DP WORKLOAD. FEDSIM had already determined that the functional user requirements from the XYZ system resembled a typical database management environment. Further discussions with the user community assisted in the development of a set of functional transaction definitions to characterize the production XYZ workload. The following transaction types were identified: (1) process control; (2) inquiry/display; (3) simple update; (4) complex update; or (5) complex computation. Typical definitions for these transaction types are shown in Table 1.

Table 1. XYZ Transaction Definitions

PROCESS CONTROL:	FUNCTION SELECTION MENU REQUIRES NO ACCESS OF DATABASE
INQUIRY/DISPLAY:	SYSTEM PROMPTS FOR INFO QUALIFIERS SEARCHES DATABASE AND RETRIEVES INFO
SIMPLE UPDATE:	REQUIRES USER INPUT TO SCREEN FIELDS UPDATES DATABASE
SIMPLE COMPUTATION:	REQUIRES USER INPUT DATABASE SEARCH PERFORMS CALCULATION ON DATA RETRIEVED
COMPLEX COMPUTATION:	MORE EXTENSIVE SEARCH MORE COMPLEX CALCULATION REQUIRED

These transaction definitions derived from the workload translation process allowed the XYZ workload to be characterized by transaction type, volume, and estimated system resource requirements. These results are described in the following section.

3.3.3 HOW MUCH WORK. FEDSIM developed a profile of the XYZ workload using the major workload components identified by the workload regression model and the transaction types described above. The linear regression technique identified the significant collection activities related to XYZ and estimated model coefficients of all explanatory variables for the XYZ workload. These coefficients were used with "information screen" estimates for each XYZ function to estimate the total number of transactions for a yearly XYZ workload at each region. These information screen estimates resulted from detailed discussions with users and system designers of the XYZ system. The number of screens and data content for each major XYZ function were identified. FEDSIM assumed that these identified screens for the accounts management activities could also be defined as "XYZ transactions". The basis for this assumption was logical since even the process of scrolling through screen displays at a user terminal requires some interaction with user communications software at the host system, hence an interrupt and some amount of host system processing. Therefore, the mathematical product of the model coefficients and the screen totals provided transaction subtotals for each major accounts management activity. The resultant sum of these transaction subtotals was the estimated total XYZ transaction volume for each XYZ location during a one year period.

FEDSIM also estimated the distribution of transaction types at each XYZ location. From the worksheets previously discussed, FEDSIM calculated the total number of transactions in each transaction type for all major accounts management activities. These figures, when applied as a ratio to the total transaction figures, provided percentiles for each transaction type at each XYZ location. Table 2 shows

the distribution of the XYZ workload by transaction type for each of the major XYZ tasks.

Table 2. XYZ Workload Distribution by Transaction Type

TRANSACTION TYPE	XYZ PROCESS[1] UC	INS1	INS2	INV	PENALTY	TOTAL
Process Control	28.0	28.7	28.7	30.3	40.0	30.4
Inquiry/Display	47.0	32.1	32.1	57.1	51.6	46.9
Simple Update	20.0	29.0	29.0	10.5	8.4	17.6
Simple Computation	2.6	8.0	8.0	2.1	-	4.2
Complex Computation	2.4	2.2	2.2	-	-	0.9
TOTAL	100.0	100.0	100.0	100.0	100.0	100.0

[1] Percent of task by type.

FEDSIM found that process control (30.4%), inquiry/display (46.9%), and simple update (17.6%) would probably be the major transaction types. These figures represent 95 percent of total XYZ processing. The proportions of total transactions represented by each transaction type are also shown in the table.

The XYZ transaction types were then defined in terms of their estimated resource requirements. FEDSIM derived these definitions by conducting a survey of large IBM installations that maintain database management systems (DBMS's) approximating the XYZ system in size and workload. The results of FEDSIM's survey are summarized in Table 3. Note that the resource requirements for each transaction type indicate a best/worst case range. Due to time limitations and to the variety of commercially available DBMS's, FEDSIM reported only an approximate range. Improved accuracy is possible only after XYZ database design and implementation plans are finalized.

Table 3. XYZ Transaction Resource Definitions

TRANSACTION TYPE	RESOURCE REQUIREMENTS CPU TIME (SECONDS)[1] MINIMUM	MAXIMUM	DISK I/O MINIMUM	MAXIMUM
PROCESS CONTROL	.005	.010	2	5
INQUIRY/DISPLAY	.040	.080	5	20
SIMPLE UPDATE	.300	.360	5	20
SIMPLE COMPUTATION	.200	.500	10	30
COMPLEX COMPUTATION	.600	1.130	30	100

[1] CPU estimates based on IBM 3081K.

FEDSIM used these workload distributions and transaction definitions to determine the total XYZ transaction volume and resource requirements for each XYZ location. The 1986 manual workload volumes, transaction distributions, and resource usage definitions were the basis for these estimates. These data are shown in Table 4. In practice, the capacity planner should update the distributions to derive new workload sizing estimates as more complete data become available.

Table 4 shows the estimated 1986 XYZ workload volumes. Transaction totals are estimated for a one-year period. Assuming a normally distributed arrival pattern, a 40-hour processing week, and a 50-week year, FEDSIM also estimated the average hourly workload.

The Los Angeles region was predicted to have the largest estimated workload, with over 41.4 million transactions for the year, or 20,700 transactions per hour. The average XYZ system would process approximately 24 million transactions during a year or 12,000 per hour.

Table 4. XYZ Workload Estimates, 1986 Base

XYZ LOCATION	TOTAL TRANSACTIONS[1]	TRANSACTIONS PER HOUR[2]
Baltimore	11.2	5,600
Boston	20.8	10,400
Philadelphia	11.9	6,000
Buffalo	15.9	8,000
New York	27.4	13,700
Newark	14.0	7,000
Houston	37.6	18,800
Dallas	31.0	15,500
Denver	31.1	15,500
Seattle	31.1	15,600
San Francisco	32.3	16,200
Los Angeles	41.4	20,700
Jacksonville	26.2	13,100
Atlanta	25.8	12,900
Nashville	21.2	10,600
Indianapolis	17.4	8,700
Cleveland	20.1	10,100
Detroit	22.9	11,500
Chicago	26.4	13,200
St. Louis	11.8	5,900

[1] Millions.
[2] Assumes normally distributed arrival.

3.3.4 What Size System

3.3.4.1 Host Processor Size

To determine the "class" of machine required for processing the XYZ workload, FEDSIM developed an analytic model using BGS System's CRYSTAL and BEST/1 products. Analytical models are structured mathematical relationships used to describe the dynamic behavior of hardware and software components of computer systems. These models can estimate the performance of computer systems and are often used to predict the data processing requirements for newly-designed systems. They are especially useful for comparative analyses since changes to workload descriptions and hardware configurations can be done relatively quickly and easily. The agency was particularly interested in (1) estimating the "class of machine" (i.e., size) necessary to support XYZ at each location and (2) using the flexibility of analytic models to evaluate the effects of environmental changes (i.e., "what if" analyses).

Using the analytic model to relate the resource estimates to the workload volumes, FEDSIM estimated the CPU time and disk I/O requirements. These statistics are shown in Table 5. FEDSIM also estimated ranges of CPU busy and disk I/O rates. The CPU estimates assume an IBM 3081K processor, a 40-hour processing week, and a 50 week year.

The Los Angeles region would have the largest estimated resource requirements. The 41.4 million transaction workload would consume approximately 1,000 to 1,600 CPU hours during a year. An IBM 3081K processor would be utilized 52 to 80 percent during prime shift. The Los Angeles workload would also perform between 189 and 691 million disk I/O's, or between 26 and 96 I/O's per second. The average XYZ system would require 573 to 842 hours of CPU time and 108 to 395 million I/O's per year.

Table 5. XYZ Resource Estimates, 1986 Base

XYZ LOCATION	CPU ANNUAL (HR)[1]	CPU USAGE (%)[2]	DISK I/O ANNUAL (*10^6)	DISK I/O PER SEC[2]
Baltimore	273 - 425	14 - 21	51 - 186	7 - 26
Boston	469 - 733	23 - 37	92 - 340	13 - 47
Philadelphia	274 - 428	14 - 21	53 - 196	7 - 27
Buffalo	378 - 588	19 - 29	72 - 263	10 - 37
New York	549 - 866	27 - 43	118 - 436	16 - 61
Newark	336 - 522	17 - 26	63 - 231	9 - 32
Houston	844 - 1,319	42 - 66	167 - 614	23 - 85
Dallas	751 - 1,167	38 - 58	140 - 513	19 - 71
Denver	777 - 1,205	39 - 60	142 - 519	20 - 72
Seattle	773 - 1,199	39 - 60	142 - 519	20 - 72
San Francisco	759 - 1,184	38 - 59	145 - 534	20 - 74
Los Angeles	1,031 - 1,600	52 - 80	189 - 691	26 - 96
Jacksonville	640 - 994	32 - 50	119 - 435	16 - 60
Atlanta	626 - 974	31 - 49	117 - 429	16 - 60
Nashville	538 - 834	27 - 42	97 - 355	13 - 49
Indianapolis	415 - 646	21 - 32	78 - 287	11 - 40
Cleveland	512 - 792	26 - 40	92 - 336	13 - 47
Detroit	559 - 868	28 - 43	104 - 380	14 - 53
Chicago	642 - 999	32 - 50	120 - 438	17 - 61
St. Louis	311 - 480	16 - 24	55 - 199	8 - 28

[1] IBM 3081 CPU hours.
[2] Based on 2,000 prime shift hours per year.

3.3.4.2 On-line Storage Requirements

FEDSIM reviewed the XYZ database design documentation provided by the agency. This documentation provided the current database schemata (i.e., logical file organization, file element descriptors, record and field lengths) under development for XYZ. FEDSIM then interviewed agency XYZ project personnel to identify the relationship of the XYZ workload (i.e., accounts receivables + investigations) to the schemata. These relationships effectively became numerical multipliers to determine the relative file sizes within the database. Then, the sum of the relative file sizes became the estimated total on-line storage capacity (in bytes) to accommodate the XYZ database at each location. However, one must recognize that this statistic represents the storage capacity required if the XYZ database is allowed to grow unmanaged and uncontrolled during a complete XYZ production year. The frequency and extent of database management practices, such as reorganizations, archivals, etc., as well as XYZ workload arrival patterns, will have a dramatic effect on the on-line storage requirements at any specified time during an XYZ production year.

FEDSIM estimated the size of the XYZ database in the following manner. The agency's database design documentation identified the types and lengths of records in various files comprising the XYZ database. XYZ project team personnel also provided several other records and corresponding lengths not identified in this documentation that were being considered for the XYZ database.

Agency personnel estimated individual file sizes by projecting the number of occurrences for each type of record based on various criteria. These criteria varied among the record types and were developed based on the experience of the XYZ project team personnel working with FEDSIM on this portion of the effort. These criteria established algorithms for determining the number of records in specific files and for estimating the size of related files in the database structure. FEDSIM developed a spreadsheet using LOTUS 1-2-3 to algorithmically size the total database at each XYZ location. This sizing depended on the actual 1986 manual workload, the estimated workload from the regression model, and the estimated number of accounts at each site. The total XYZ database size was the sum of the lengths of all record types included in the XYZ database or resulting from XYZ processing, multiplied by the number of occurrences per year of each record type (these statistics were estimated by the agency personnel).

The agency was not certain how database archives would be accomplished. However, FEDSIM decided that one of three alternatives would probably be used: (1) data written to an on-line archive file but not removed from the active database; (2) data written to an on-line archive file and removed from the active database; or (3) data written to an off-line (e.g., tape) archive file and removed from the active database. Option 1 would require the most on-line storage and is not practical unless an archive backup is required. The major difference in storage requirements due to the archive file depends on Option 2 or Option 3. With Option 2, the size of the archive file essentially equals the storage required to maintain the data in the active database (i.e., the on-line requirement stays the same with or without an archive file). With Option 3, the on-line storage requirement decreases by the size of the archive file.

The XYZ database size estimates are shown in Table 6. This table shows two size estimates in bytes for each region. One estimate indicates the database size

assuming Option 2 above; the second estimate assumes Option 3. For purposes of discussion, a single IBM 3380 disk can store 2.52 gigabytes of data, or over two and a half billion bytes of data. Therefore, the storage estimates indicate that each XYZ site would require the storage equivalent to one to two 3380-size disks. (Of course, to ensure good performance, a database would be spread over several partially-filled disks on separate channels and controllers.)

Table 6. XYZ Database Storage Estimates, 1986 Base

XYZ LOCATION	DATABASE STORAGE (GB) NO ARCHIVE[1]	OFF-LINE ARCHIVE
Baltimore	1.45	1.28
Boston	1.69	1.47
Philadelphia	2.55	2.25
Buffalo	2.00	1.77
New York	3.55	3.14
Newark	1.91	1.70
Houston	4.57	4.04
Dallas	3.77	3.35
Denver	3.78	3.34
Seattle	3.75	3.29
San Francisco	3.65	3.19
Los Angeles	4.56	4.01
Jacksonville	3.12	2.78
Atlanta	3.10	2.69
Nashville	2.55	2.28
Indianapolis	2.26	2.02
Cleveland	2.56	2.27
Detroit	2.88	2.51
Chicago	2.97	2.60
St. Louis	1.52	1.36

[1] With archive file, if data compression occurs.

4. SUMMARY

This paper provides a structured approach to addressing capacity questions regarding unknown workloads. The usefulness of this technique should be viewed with regard to the risks associated with the results. The sizing methodology presented herein can be summarized by the following four steps:

(1) Define the workload in NBU's;

(2) Translate the NBU's to data processing work units;

(3) Estimate the workload volume;

(4) Estimate the resource requirements.

Perhaps, the most significant benefit from this approach, as echoed by the case study agency, was that the systematic method highlighted the system design factors contributing to the automated workload, identified the previously unquantified workload factors, and provided techniques and direction for further analyses. With this analytic structure in place, the sizing model can undergo continual refinements and validations up to the point of hardware acquisition.

AN EXPERIMENT IN THE CALCULATION OF CAPTURE RATIOS

John F. Temperly
and
Robert J. Hardman
National Advanced Systems

ABSTRACT

Several activities undertaken by Capacity Managers demand a knowledge of local, site, CPU complex, and workload subset specific capture ratios. Many organizations have no way of determining other than system-wide capture ratios.

This paper discusses an experiment conducted to find a means of facilitating the calculation of capture ratios for individual workload groups. These calculations have been based on data readily and commonly available from a standard MXG Performance Data Base.

Comments are made on the validity and effectiveness of some methods of workload group capture ratio calculation and on a means of establishing a prima facie accuracy for the results obtained.

The evaluation pays particular attention to the multiple linear regression method of workload group capture ratio calculation recommended by Dr. Barry Merrill in his book relating to MXG.

1.0 INTRODUCTION

1.1 MEASUREMENT OF UTILIZATION AND PERFORMANCE

In the IBM MVS operating system environment, manuals and other capacity management literature provide considerable discussion of the shortcomings of software tools which measure system utilization and performance.

One common theme is the inability of those software tools to account for the CPU utilization.

Total CPU (busy) time must be calculated by taking the difference between the elapsed time of the measurement interval and the wait time accumulated for the CPU complex during that interval. The full, accurate measurement of time consumed in processing individual subsets of the total workload is rarely accomplished and reported. Typically, the Sum of the times recorded for all subsets is less than Total CPU time calculated.

1.2 THE CAPTURE RATIO CONCEPT

The concept of a Capture Ratio is simple and may be familiar. The ratio of the Sum of workload subset CPU times to the Total CPU time, defines the System-wide Capture Ratio.

The aim of the experiment documented here was to find a satisfactory method of calculating capture ratios for workload subsets. Such methods involve adjusting subset CPU times by distributing the overhead of the system (the difference between Sum and Total CPU times) across the workload subsets. The ratio of the CPU time measured for a subset to the adjusted subset time, defines the Capture Ratio for that subset of the workload.

1.3 CAPTURE RATIO VALUE VARIATION

There is a second theme common in the literature. It is that Capture Ratios for any workload will vary under different measurement circumstances.

Each CPU product which is an implementation of the standard application compatible architecture, uses its own numbers or sizes of CPU architectural components. These include the number of instruction processors, cache buffer and main storage sizes, or uniquely microcoded performance enhancements.

Operating system versions not only make use of or manage the CPU sources differently, but also create and deal with overhead processing in their own way.

Other program product software changes, for example, to compilers, major application subsystems such as CICS, or the introduction of 4GL's or a DBMS to reimplement the processing of the same functional workload, will result in different overhead demands on the operating system.

Most importantly perhaps, the measurement and accounting tools are constantly being enhanced. Their ability to measure CPU consumption generally has improved with each new version. Version changes have often brought with them redefinition of some of the quantities measured or perhaps a shift in the method of measurement.

Any differences in these system features between CPU's used to accomplish processing, or any changes to them during the life of a particular CPU, may impact the capture ratio for a given workload.

2.0 REASONS FOR CALCULATING WORKLOAD GROUP CAPTURE RATIOS

There are at least three reasons why capture ratios for individual subset workloads might be important.

2.1 WORKLOAD FORECASTING

Probably the most common is the need to establish a reasonably accurate base on which to build estimates of CPU consumption for forecast workload levels. Subsets of the workload not only exhibit differences in forecast growth patterns, but also in their characteristic transaction profiles and the CPU power consumed in processing their typical transactions. For existing applications, use of workload subset specific Capture Ratios in adjusting reported CPU utilization will produce more accurate estimates of the basic quantum of CPU consumption computed for each element of the business workload.

2.2 CPU PERFORMANCE COMPARISONS

CPU performance comparisons at the time of evaluating bids from hardware suppliers, or for acceptance testing purposes to satisfy contract conditions, are becoming quite complex. Many of the comparisons involve the calculation of Internal Throughput Rate (ITR) figures. These figures are based on the processing of a specific subset of the total production workload executed by the machines being compared. The ITR calculations require a knowledge of the Capture Ratio(s) relevant to the specific workload subset for each CPU complex. Only by adjusting CPU resource consumption figures for a specific workload using the related Capture Ratio, can valid ITR comparisons be made between CPU complexes.

2.3 CHARGEOUT

The frequency of CPU power upgrades is increasing with the rapidly increasing demand from users for that power. The occurrence of dynamic allocation of work among the individual CPU complexes in a service network may also be increasing. Organizations experiencing relatively frequent shifts of workload from one CPU to another may have difficulty in maintaining consistency in billing for CPU time. It is possible that chargeout systems which use CPU and workload subset specific Capture Ratios to adjust system accounting records for individual jobs' CPU consumption will produce more consistent bills for a given volume and type of work.

3.0 "CLASSICAL" APPROACHES

3.1 LITERATURE SEARCHES

As part of the preparation for and continuing review of the experiment in the calculation of Capture Ratios, searches of available literature were conducted. Potential sources searched included various manuals from IBM in particular, and computer held databases, including NAS' SPIRE system, IBM's INFOSYS, and several databases of abstracts available through the Australian National Library.

In all searches, care was taken to narrow down the volume of "hits" by requesting that only those references which specifically dealt with the calculation of the ratios were returned. Very few references were returned as a result. No attempt has been made to determine whether the search criteria were, in fact, too narrow, or whether there is simply not a lot written on the subject.

A clear pattern emerged in spite of the low yield from the searches. It could be said that two classical approaches to capture ratio calculation for workload subsets exist. The first is what might be termed the heuristic approach, while the second is based firmly on the use of multiple linear regression in the manner proposed by Merrill, mentioned earlier.

3.2 APPORTIONMENT HEURISTICS

Heuristic methods proposed included time-consuming trial and error, starting from a set of published ratios and adjusting them until a good consistent fit is found for the measurement data for the user's own site. No reference described a successful attempt to use this method, nor was any useful level of detail for unsuccessful attempts revealed.

Other common heuristic methods are the variations on the theme of apportionment. This approach calls for the overhead or time not "billed" to any specific workload subset, to be allocated to the subsets in proportion to the consumption of a known measured resource.

IBM's SMF/RMF software provides a breakdown of time accounted for each performance group for an MVS system as SRB and TCB or CPU time components. The usual method of apportionment of overhead time is in terms of the relative amounts of SRB time recorded for the groups. This method is based on an assumption that overhead work performed by the CPU is likely to be related to requests made for operating system services by user tasks. Such a method has an appealing simplicity and readily permits the retention of performance groups as the workload subsets under consideration.

Alternatives for the basis for apportionment come to mind, e.g., the ratio of SIO's attributed to the different workloads.

3.3 MULTIPLE LINEAR REGRESSION

The general form of the multiple linear regression equation is:

$$Y = b_0 + b_1 X_1 + b_2 X_2 + ... + b_n X_n + e$$

In the Capture Ratio calculation:

Y is the total CPU resource consumption during a measurement interval (total active or busy time),

X_i represents CPU resource actually measured for a specific workload (part of the CPU busy time),

b_0 is the intercept,

b_i is the estimate of the true regression coefficient which describes the relationship between total CPU resource usage and that of the individual $workload_i$, and

e is an error term.

In an ideal world, each of the b_i's would have the value 1.0, all of the total resource consumption being accounted for by one workload subset or another. In the real world, this is not the case. Theory associated with the notion and phenomenon of Capture Ratios says that the regression coefficient estimates (b_i) should all be positive, should all be greater than or equal to 1.0, and in fact b_i is the reciprocal of the Capture Ratio for $workload_i$.

4.0 AVAILABLE TOOLS

At the outset, it was recognized that the BGS Systems package, CAPTURE/MVS, has gained acceptance as a tool which provides a source of the information required. However, there is a need for a low- or no-cost alternative which provides an interface to other commonly available capacity management software.

Many Capacity Managers are becoming quite familar with Merrill's Expanded Guide to Computer Performance Evaluation (MXG) and the SAS based package of routines that both SAS Software and Dr. Merrill market as an aid to performance analysis.

MXG is the basis of one of two major components in the NAS Capacity Tool used by NAS System Engineers as a support aid for customers wishing to profile the performance or resource utilization of their computing systems, or to model systems to predict performance. Neither the Tool nor the standard MXG code includes a routine for calculating workload specific Capture Ratios.

5.0 THE DO-IT-YOURSELF MXG EXTENSION

It seemed reasonable, given the requirements, the environment and available tools, to use Dr. Merrill's ideas and the MXG portion of the Tool as a starting point for developing a DIY Capture Ratio calculation routine for subset workloads. Consequently, a SAS program was written to interface with an MXG generated Performance Data Base (PDB) to implement both an apportionment approach and the multiple linear regression method of determining the ratios.

The apportionment was affected by the allocation of overhead time according to the relative amount of SRB time recorded for a performance group. SAS' PROC REG was used as the regression routine.

Data available as input to the SAS program were PDB's built from samples of SMF/RMF data from two NAS customer sites in Australia and from a third site in the USA. Standard MXG code was used in the reduction of the SMF/RMF data to an MXG PDB. As much detail as possible was retained by leaving unchanged the workload subset groupings and using the sites' performance groups as subset definitions.

6.0 INITIAL RESULTS

The initial attempts at calculating workload specific Capture Ratios gave rise to some quite unsatisfactory, if surprising results.

Initial executions of the SAS routine which analyzed data from an NAS Australia customer, produced valid ratios from the apportionment, negative regression coefficients for some workloads (performance groups) and rather large magnitude intercepts (b_0's). The negative coefficients were a particular concern as in the context of the "theory" of calculating the ratios via regression, a negative coefficient or one less than 1.0 has no validity. An intercept term of any size required interpretation.

At this point, it was decided to examine the statistical validity of the method. Statistical and mathematical consultants SIROMATH Pty. Limited were asked for an opinion. Also, discussions took place directly with Dr. Merrill as well as with representatives of International System Services Corp. (ISS), purveyors of ISS-THREE, a Capacity Planning and Modeling Package.

7.0 FEEDBACK FROM DISCUSSIONS WITH EXPERTS

7.1 SIROMATH COMMENTS

SIROMATH went so far as to suggest that the use of multiple linear regression might be invalid for any number of reasons. Their reasons included the possibilities of autocorrelation between workloads and of a time variation in the nature of any relationship between total and specific workload related CPU resource consumptions.

While this may well be true, it was decided at the time not to mount a project with SIROMATH to find the best method, given that they admitted to being unfamilar with the physical world aspects of the problem. A fully rigorous approach to the statistics was seen to be unwarranted as a pragmatic approach is all that is required.

7.2 DR. MERRILL'S COMMENTS

Discussion with Dr. Merrill threw a good deal of light on the subject. It should be noted that from a limited number of observations, attempts were being made to find a regression relationship between total CPU resource consumption and resource consumption for 17 performance groups in an MVS environment. Also, attempts were made to force a regression without an intercept term, an optional feature of SAS' PROC REG, basically because it was believed that the whole idea was to apportion all overhead to individual (application/user) workloads and consequently an intercept had no meaning or place in the equation.

The following are the main comments and suggestions made by Dr. Merrill in an initial discussion:

a. There is no need to remove the intercept term from regression analysis of Performance Groups (PG's) CPU time consumption to determine Capture Ratios.

b. The intercept represents work going on in a system with all (user) PG's "at rest", i.e., the MVS dispatcher, SRM, RMF, etc. are all simply cycling around looking for things to do, or measure, or on which to report.

c. The intercept value should be small and represent a low percentage of the total work and CPU consumption in a busy system, otherwise:

i) a performance group of significance has been missed or;

(ii) in deliberately leaving out PG's not of prime interest from the equation, too much has been attributed to the idling system and probably to the error term as well.

d. Use observations from Peak Hours only, especially in non-XA systems.

e. It is necessary to combine like workloads, e.g., PG's of all JES nodes or all TSO response periods, into a logical PG or workload type group for purposes of the regression.

f. Plot measurements of each PG's resource consumption against time, using SAS, to assist in removal of outliers. They should be easy to pick visually.

g. PG's with bi-modal workloads:

(i) will show up in graphs, and
(ii) will give strange results.

h. Typically, in performing the calculations to derive capture ratios, Dr. Merrill has used 30 days data with 6 observations from the peak periods of each day.

After making many of the changes Dr. Merrill suggested, it was possible to go through with him the amended code and the results it was producing at CMG in Las Vegas. At that time, he made further suggestions, mainly concerning the graphical presentation of the initial data. The changes were to permit IEBIBALL processing to identify potentially troublesome outlier observations. He also recommended the use of SAS to plot the residual values (actual-predicted) of total CPU resource consumption, based on the sets of Capture Ratios calculated by the routine.

7.3 COMMENTS FROM ISS

Discussions with the ISS personnel were brief and aimed at determining whether their package attempted to calculate Capture Ratios and, if so, by what method. ISS-THREE does attempt the calculation by apportioning unassigned overhead to specific workload groups in accordance with the ratio of the reported SRB resource consumption accounted for by the workloads.

8.0 DATA RELATED ISSUES

8.1 NUMBERS OF OBSERVATIONS

It has been suggested that a rule-of-thumb (ROT) for regression analysis of data similar to that processed in the experiment is that there should be between 10 and 20 observations for every independent variable in the equation. The thought behind this ROT is that a certain minimum number of observations is required before regression analysis will provide any worthwhile result, and that the numbers of observations suggested are necessary if analysis is to have any chance of determining properly whether the established regression relationship is statistically significant.

8.2 GROUPING OF OBSERVATIONS

Following Dr. Merrill's suggestions, it was decided that an adequate grouping of observations into more general workload subsets was required. Data from PG's was grouped into five subsets:

-BATCH
-TSO
-CICS
-DBMS
-OTHER

In terms of the ROT stated above, this meant that the data available contained quite sufficient observations for a regression analysis to be carried out satisfactorily and for its statistical significance to be determined. At the same time, it meant that the group to which observations for a particular PG should be allocated was not always clearly defined. This resulted in some arbitrary assignments which may impact results in unpredictable ways and could be a factor in the unsatisfactory results obtained from analysis of the data from the U.S. site (see below).

There is a message here for Capacity Managers they have a role in helping to determine the MVS system's Performance Groups to facilitate their work as well as the dynamic management of system performance or system resource accounting.

8.3 REMOVAL OF OUTLIERS

The SAS routine developed produces six sets of calculated ratios. These sets are obtained by the different methods of calculation of Capture Ratios:

a. Using "unfiltered" data

(i) regression with intercept
(ii) regression without intercept
(iii) apportionment of overhead (ratio of SRB basis)

b. Using filtered data

(i))
(ii)) as above
(iii))

Two approaches to data filtering to remove outliers have been tried. In the first, observations are retained only if the CPU time values are within two standard deviations above the mean. For each observation, measured values are compared with

the mean as a lower threshold and

mean-plus-two-standard-deviations
as the upper threshold

for total CPU time and for each workload group CPU time. The aim here is to preserve a very homogeneous set of observations made at times of relatively peak resource consumption.

The alternative method retains a relatively high proportion of observations where this is important to provision of greater significance to the regression relationship achieved. It does this by preserving those observations with values within one standard deviation either side of the mean (in all instances, total and individual workload groups). Again outliers are removed, improving the homogeneity of the data.

8.4 REPEATABILITY AND CONSISTENCY OF RESULTS

Several references and personnel from the U.S. site mention the need for consistency in results obtained from Capture Ratio calculations. No attempt has yet been made to incorporate any consistency check into the SAS program. On hearing of the experiment, the capacity manager at one site requested a copy of the SAS program to perform some calculation tests himself. Early results based on hourly observation data collected during prime shift were mixed and inconsistent. After twenty days' data (180 observations) became available, however, the results showed a marked improvement in consistency as well as validity. At the time of writing, these tests had continued with up to 40 days data with little variation in the Capture Ratio values calculated after the twentieth day.

9.0 ATTEMPTED VALIDATION AND EVALUATION

9.1 VALIDITY

The SAS program developed has been tested to the point that it works in some circumstances, producing valid ratio values from some PDB data, i.e., positive values between 0.0 and 1.0.

9.2 A CRITERION FOR GOOD RESULTS

The program provides a means for a quick evaluation of which set of calculated Capture Ratios appears to be best - best being defined as the set which results in the smallest range of residual values, depicted graphically.

In testing carried out to date, the best results from the available PDB data have been obtained by regression with intercept, using filtered data.

9.3 ACCURACY - A BENCHMARK

It has been revealed that personnel at the U.S. site from which PDB data was obtained have been calculating workload specific Capture Ratios for their CPU complexes for some years. Their method was not described in detail, but it was claimed that it is based on a trial and error heuristic, calculation being complete when weekly analysis results are stable over a period of about two months for a given complex, operating system version, and microcode level. A curious fact is that the results that they were prepared to share included b_i values, for all the world like estimates of regression coefficients.

These U.S. site results were an obvious benchmark against which to gauge the accuracy of the results obtained from the developed SAS program. The routine has yet to produce results that are close enough to those quoted by the U.S. site to satisfy the author. The results of processing unfiltered data from the site are invalid giving Capture Ratios greater than 1.0. Filtering the data yielded even more unsatisfactory results as the process left a mere six observations on which to base calculations.

It is clear from examination of the program's graphical output that the U.S. site data which provides one observation for every quarter hour prime shift period for a week, is far from homogeneous. Data "conditioning" issues need to be resolved before the experiment can be considered to have been completed.

10.0 CONTINUING THE EXPERIMENT

The experiment has life in it yet if it is to reach the desired conclusion of consistently producing valid and accurate workload subset Capture Ratios from the widest possible range of data processed. The SAS program developed needs more testing to overcome the impact of poor data conditioning on its results and a calibration of the results against those of accepted packages needs to be performed.

It seems obvious that a comparison of the SAS routine's Capture Ratio values with those produced from the same data by either CAPTURE/MVS or ISS-THREE would be of value. This will be undertaken. A site possessing one of those tools has been identified and agreement reached for SMF/RMF data from that site, together with results produced by that package, to be made available for analysis and comparison.

11.0 CONCLUSION

Already the project has produced results which are of value in workload forecasting, to CPU complex performance comparisons, and possibly to chargeout calculations.

The project might also provide insight to others who have seen the need to embark on the exercise of calculating workload subset Capture Ratios for themselves. It will certainly be of value in pointing out some of the issues that need to be addressed in such an undertaking.

12.0 ACKNOWLEDGEMENTS

The author is indebted to many people for their contributions to the progress of the experiment or their forbearance while it proved to be a consuming, if part-time obsession. In particular, acknowledgement must be made of the assistance received from Bob Hardman for acquiring the U.S. test data and his encouragement to continue with the experiment. Dr. Bruce Howarth provided some suggestions for incorporation into the program and a critique of an early draft of this paper. Dr. Barry Merrill contributed several suggestions during our discussions and markedly improved the author's insight into the problem. Bill Gibson was patient with a relative novice. He demonstrated some clever techniques for including conditionally interpreted code into SAS programs, thus overcoming some data processing generalization difficulties. The author's wife was the most forbearing.

13.0 REFERENCES

1. Csontos, E.: Workload Prediction Using Time-Series Analysis. Computer Performance (GB) Vol. 5, No. 2, 70-9 June 1984

2. Felix-Simpson S.: Using CPU Measurement Data for Capacity Planning: How to Calculate Capture Ratios. CMG XV:International Conference on the Management and Performance Evaluation of Computer Systems. Conference Proceedings 591-3, 1984.

3. Chapman,L.: Using PROC NLIN to Develop Capture Ratios. SUGI Paper, 1987.

4. Major,J.B.: Capacity Planning With A SIO Methodology. SHARE 65 Conference. New Orleans 1985. Conference Session M675, Project CME.

5. Merrill,H.W.: Merrill's Expanded Guide to Computer Performance Evaluation Using the SAS System. SAS Institute Inc., Cary, NC, USA, 1984.

6. Radecki,K.L.: Introduction to Large Processor Capacity and Performance Evaluation. Technical Bulletin GG66-0232-00. IBM Washington Systems Center, May 1986.

14.0 APPENDIX

Capture Ratios Quoted or Calculated.

Source	When/OS	Batch	TSO	CICS	DBMS
Major	1980 MVS(SE?) (UP)	.85	.68	.80 (CICS/IMS)	
Major	1980 MVS(SE?) (MP)	.75	.60	.70 (CICS/IMS)	
U.S. Site	1986 MVS/XA	.99	.66	--	.85 (IMS)
Australian Site$_1$	1987 MVS/SP	.82	.88	.46	-- (*)
Australian Site$_2$	1987 MVS/XA	.85	.84	.94	-- (*)

(*) Calculated using the SAS program developed as part of the experiment.

THE USE OF REGRESSION ANALYSIS IN THE CALCULATION OF CAPTURE RATIOS

Mel F. Kirshin
C-I-L Inc.

ABSTRACT

One of the most basic technical issues faced by all capacity planners is the calculation of capture ratios; however, most capacity planners will use rules of thumb or resort to hit and miss iterative methods when it comes to this ultimately critical statistic.

Although regarded by many practitioners as a dangerous method, when used correctly it will yield reliable results. This paper presents C-I-L Inc.'s experience with regression analysis (SAS PROC REG) to calculate capture ratios.

1. BACKGROUND

1.1 CAPTURE RATIOS

Anybody who tracks and forecasts load needs capture ratios. This is because you normally track load by workload categories, e.g. TSO, Batch Production, IMS etc. By assigning unique performance group numbers to these workload groups, you may obtain TCB and SRB CPU time from Type 72 RMF records. However, there is a significant amount of overhead CPU which is consumed by the system and not reported in Type 72's. The total CPU time consumed is, however, reflected in Type 70 RMF records. The ratio of Type 72 "captured" CPU time to the total CPU time consumed on behalf of any given workload is called Capture Ratio.

Regression analysis may be used to determine the capture ratios.

1.2 REGRESSION ANALYSIS

Regression analysis is an analytical process by which a line is fitted through a number of points. In the case of a straight line, it is referred to as simple linear regression. If a good fit can be established, then the equation of the line can be used to predict the behavior of the points.

Similarly, in the case of points plotted in 3-dimensional space, a plane may be fitted to the points. This is referred to as multiple linear regression, and is the kind we use in connection with capture ratios.

1.3 REGRESSION ANALYSIS AND CAPTURE RATIOS

How does a plane relate to capture ratios? Consider an environment with two workload categories - Wkld A and Wkld B. Points (SAS(r) observations) for regression can be obtained by breaking the load down into intervals of 15 minutes to 1 hour. Each observation must contain the variables A, B, and C representing the captured CPU time per interval of Workloads A and B and the total CPU time per interval for the system respectively.

We can take several observations and plot them in 3-dimensional space as shown in Figure 1. The values for A, B and C are plotted on the X, Y, and Z-axes respectively. By using multiple linear regression, we can fit a plane through the points. The SAS(R) PROC REG is coded as follows:

```
PROC REG;
   MODEL C = A B ;
```

The output of this procedure is shown in Figure 2. The parameter estimate column (2) gives the intercept term as well as the coefficients for variables A and B. The equation of our plane is therefore:

$$T = 1.21\ A + 1.15\ B + .44$$

The constant term, or intercept, represents the average residual CPU time accruing during the interval. In this case an interval size of 15 minutes was chosen. Our residual is then .44 seconds in 15 minutes - a negligible amount. When the intercept is this small, the capture ratios of A and B are simply the inverse of their respective coefficients in the equation of the plane. The capture ratio for workload A

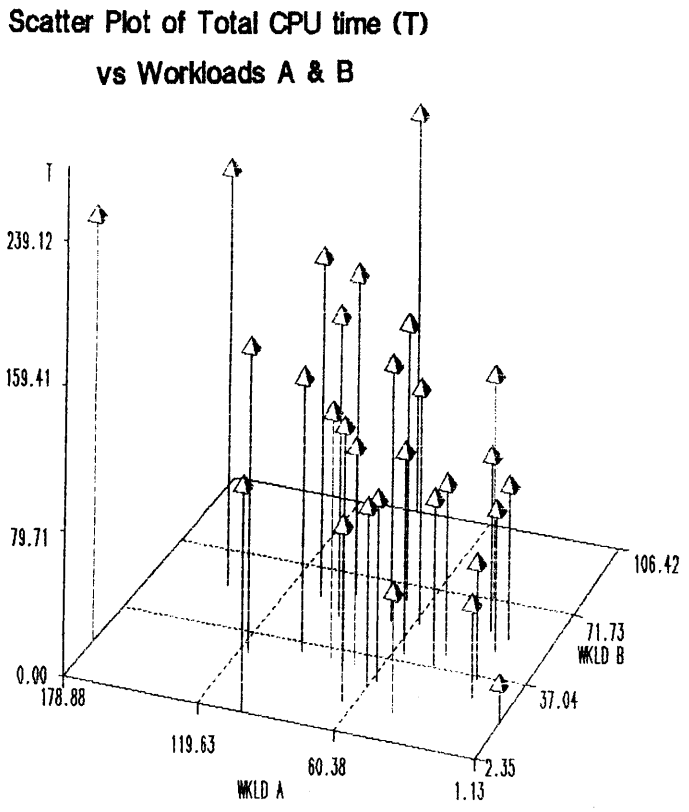


Figure 1

DEP VARIABLE: T

ANALYSIS OF VARIANCE

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	2	82764.29276	41382.14638	999999.990	0.0001
ERROR	26	0.07170794	0.002757998		
C TOTAL	28	82764.36447			

ROOT MSE	0.05251664 ③	R-SQUARE	1.0000
DEP MEAN	129.329	ADJ R-SQ	1.0000
C.V.	0.04060702		

PARAMETER ESTIMATES

① VARIABLE	DF	② PARAMETER ESTIMATE	STANDARD ERROR	T FOR HO: PARAMETER=0	PROB > |T|
INTERCEP	1	0.44696994	0.02711747	16.483	0.0001 ④
A	1	1.21028310	0.000248883	4862.861	0.0001
B	1	1.14868957	0.000463034	2480.791	0.0001

Figure 2

then is 1/1.21 or .83 and that for workload B is 1/1.15 or .87. Although it cannot be visualized in more than three dimensions, the same concept is valid for any number of workload categories.

Or so goes the theory.

2. THE REAL WORLD

2.1 THE PITFALLS

In reality, things are not so simple. You will find that the intercept will often be close to 10% of the total load. Further, you may find that the parameter estimates (and hence the capture ratios) will be slightly different with each different set of observations used. The solution to the first problem is simple - force the regression through the origin. Although this eliminates the intercept problem, it aggravates the second problem of stability of capture ratios. When faced with such a dilemma, our first instinct is to take the ostrich approach - bury our heads in the sand and use the vendor's rules of thumb.

However, by using the following guidelines, you will produce numbers which are more applicable to your installation than any vendor's general purpose rules of thumb.

3. GUIDELINES

3.1 THE OBJECTIVES

We must bear in mind that capture ratios are only useful insofar as they help us achieve our objectives - Workload tracking and forecasting. Our criteria for accepting a value for any capture ratio must therefore be as follows:

(1) The capture ratios produced by applying regression analysis against a given workload, should reproduce the original total CPU load when applied to the same workload.

(2) The capture ratios determined from one set of observations should match as closely as possible those capture ratios which are produced by another set of observations from the same installation running the same workloads.

It is rather unfortunate that this will never happen. Our aim however, is to get as close as possible to this ideal.

3.2 PREPARING THE DATA

3.2.1 CATEGORIZATION OF WORKLOADS. The keyword here is HOMOGENEITY. All tasks within a workload group should either have similar capture ratios or have a stable mix (ie the proportions of tasks with different capture ratios do not change over time). Otherwise you will find the capture ratios for the offending workload group varying widely from week to week or month to month. Such volatile numbers would be difficult to use for forecasting purposes.

Another precaution is to do a separate regression for each shift, so that you end up with for example prime shift capture ratios and off prime capture ratios. Capture ratios of a particular workload group often change from shift to shift. Each observation therefore should include a SHIFT variable.

3.2.2 DATA MANIPULATION. Some manipulation of SMF data is necessary in order to provide observations in a form usable to PROC REG. In following discussion, I must assume a reasonable level of SAS expertise on the part of the reader.

Type 72's consist of one record per performance group per time interval, However, PROC REG requires that each point (or record) contain a variable for each workload group plus one for total CPU. The transformation which is required can be accomplished in SAS with the aid of PROC SUMMARY as illustrated by the following two workload example using MICS(r) data. For those unfamiliar with MICS, the SCPPGA file contains Type 72 data; PGACPUTM is the captured (TCB + SRB) CPU time; and ZONE is shift.

```
DATA;
   SET DAYS.SCPPGA01;
   DATE=DATEPART(STARTTS);
   IF PERFGRP=1 THEN WKLDA=PGACPUTM;
   ELSE IF PERFGRP=2 THEN WKLDB=PGACPUTM;
PROC SUMMARY NWAY;
   CLASS DATE ZONE HOUR;
   VAR WKLDA WKLDB;
   OUTPUT OUT=CAPTURED SUM=;
```

This code yields a SAS data set (CAPTURED) whose variables WKLDA and WKLDB contain the captured CPU times for the workloads. This SAS data set must be merged (using the MERGE statement) by time interval with Type 70 data (HARCPU file in MICS) in order to attach the total CPU time (CPUnBTM in MICS) consumed during the interval.

If a workload has a particularly small load, ie less than 10% of the average workload, then it should not enter the regression at all. You must include code in your program which will zero out all such workloads. A capture ratio of '1' will then be assigned to these workloads. This will not have a major effect on the outcome of your load tracking since you are dealing with relatively small quantities of CPU time.

3.2.3 SAMPLE SIZE / INTERVAL SIZE. The sample size is extremely important to successful regression analysis. As a rule of thumb, a PROC REG should process a data file whose number of observations is no less than 10 times the product of workload groups and number of shifts. For example, in a case of 12 defined workloads occurring in two shifts, the number of observations being processed by the PROC REG should be no less than 10 x 12 x 2 = 240 observations. If your interval size is 30 minutes, you will require data which covers at least 120 hours of wall clock time.

Interval sizes between 15 minutes and 1 hour have been tested successfully by the author. Any longer or shorter than this range, you are on your own. I suspect, however, that success is relatively independent of interval size.

3.2.4 ENVIRONMENT. All observations should be extracted from the same environment. If a configuration change such as a CPU swap or memory upgrade takes place mid-month, do not attempt to analyze the whole month in a single regression. Rather, do a separate regression for each configuration.

3.3 RUNNING THE REGRESSION

Although more repeatable results are obtained by running with an intercept (see your SAS documentation under PROC REG, NOINT), by specifying NOINT, the increase in accuracy for the period analyzed will more than compensate for the loss in repeatability of the results.

Your actual coding for the PROC REG will then be as follows:

```
PROC REG OUTEST=CR; BY SHIFT;
   MODEL T= WKLD1-WKLDn / NOINT;
```

The OUTEST= specifies a SAS file which contains the Parameter Estimates from the regression. More on this follows:

4. PROC REG OUTPUT

4.1 INTERPRETING THE OUTPUT

Referring again to figure 2, the main statistics to be concerned with are as follows:

The VARIABLE column (1) simply lists the independent variables used in the model. If NOINT is not specified, the variable INTERCEP is also included. As pointed out earlier, the column headed PARAMETER ESTIMATE (2) contains the reciprocals of the actual capture ratios calculated by the model. This is the "Bottom Line" and may be extracted via the OUTEST file. If the INTERCEP variable is present, the residual CPU time for a single interval is reflected here. The R-SQUARE (3) value shown here is the ideal, but rarely achieved in real life. This value reflects the degree to which the model (a plane in the case of two workloads) fits the data. A value close to 1 is good. The column headed PROB > !T! (4) contains values which indicate the significance of each independent variable in predicting the total CPU time, C. A number close to zero indicates high significance.

4.2 VERIFYING THE OUTPUT

Verification of the capture ratios calculated by the model must incorporate the objectives discussed in section 2.2 above.

Print out the totals of each workload group as well as the system total used in the model. Multiply the parameter estimates by the corresponding workload group totals. The result should be close (+-3%) to the system total.

Save the capture ratios calculated each time the model is run against another time period. When capture ratios computed with different sets of data from the same installation are plotted on a vertical scale of 0 to 1, a fairly level line should emerge. The workloads that show a wide change over time are not of a consistent nature and should probably be defined differently. This is particularly important if you are doing forecasting based on current data. If a capture ratio cannot hold still for real data, it certainly cannot be expected to hold still for forecasted data.

It may take some trial and error before you have your workloads defined so that the above two tests are successful. If your major loads are defined OK but some minor loads do not pass the tests, then consider the degree of error that is involved with those workload definitions. You may find that you can tolerate some "bad" results without significantly impacting the accuracy of your load tracking or load forecasting. If you find it impossible to get good results for your major workloads, regardless of how they are defined, then you will certainly have a problem tracking or forecasting with any other method, especially with vendor's rules of thumb. So do not be intimidated by your vendor rep.

CACHING THE TAPE LIBRARY
TAPE REFERENCE PATTERNS AND AUTOMATED TAPE SYSTEMS

Daniel Kaberon
Hewitt Associates

ABSTRACT

Much has been discussed about the widening gap between the power of processor complexes and the basic speeds of the peripherals which feed their appetites. Without belittling the athleticism of the average operator, how has the speed of the operator kept pace with a shop's burgeoning MIPs? What was the average tape mount-time in 1970 versus the present? In many installations, ever-enlarging tape library sizes have caused lengthier walks from a particular tape's residence to its waiting drive. Obviously, solutions to this performance shortfall are not to be made from bringing track and field coaches into our shops to consult with our operators. However the mount-time component of batch execution-time today is larger than the processor component and growing. Knowledge of a shop's fundamental tape reference patterns can be used to exploit the newly announced StorageTek 4400 automated library system as a tape library cache. This paper discusses how to analyze and characterize a shop's tape reference patterns, the issues in caching the tape library, and how to simulate a cached tape library. Results of a study of ten installations will be presented.

TAPE-REFERENCE PATTERNS

Figure 1 shows the normal duty cycle of a tape volume in a library. The tape is created by the act of its first mount. The tape may be remounted thereafter any number of times. I term the interval between creation and last mount a volume's active duty. After its last mount, the volume is reserved in the library until it expires. I term the period from last-mount until expiration, reserve duty. Finally, After the tape expires it is a member of the scratch pool until it is subsequently mounted and the cycle begins anew. By exploring the duty cycles of all the tapes in a library, we learn a great deal about a shop's tape-reference patterns.

This is a study of the use of tape. DASD usage is primarily dataset oriented. Tape is volume oriented. In comparison to DASD, there are no heavily-hit tape datasets.

Throughout this paper I refer to tapes generically. Actually there are two broad tape media categories, IBM 3480-compatible cartridges, which I refer to as Square and basic 3420-compatible tapes in a variety of densities, which I reference as Round. All analyses in this study were performed separately for Round and Square tapes. I felt this general method allowed my approaches to be valid in shops already fully converted to square as well as those with no current cartridge drives but contemplating conversion. In the former case, round tapes and their usage must be fully excluded and in the latter, round-tape usage should be fully predictive of square-tape usage. In all data analyses, the distinction between round and square was directly interpreted from the various raw data sources.

Uccel's Tape Management System's Tape Management Catalog (TMC) is a unique and comprehensive source of information about a shop's tape library. The TMC contains the status of every tape volume in a shop's library. From the TMC we can determine the creation-date and last-reference date of every tape dataset, the scratch status and what will lead to scratching he tape (i.e. expiration date, uncataloging), the tape's density and shape (round of square). We also can learn if the tape is on-site or off-site in some particular vault location. The TMC can accurately provide all the required values for quantifying the active and reserve duty periods of every tape in the library.

By calculating the days between creation and last-reference, we have quantified the active duty cycle (this is known by the SAS variable name days-to-last, DAZ2LST.) We can learn a great deal about tape reference patterns. A cumulative frequency distribution of this variable (Figure 2) shows us how many tapes have a duty-cycle less than or equal to one day, up to two days, three etcetera. From my study of nine sites, the percentage of tapes with duty-cycles less than or equal to one week ranged between 67.3% and 95.0%, mean 84.6. This tells us that the final use of a large majority of tapes is completed within a relatively short period after creation. Tape usage shows a very strong locality of reference. Put another way, if you held every tape that was created in one day, after one week only 15 percent of those tapes would ever be reference again prior to scratching. I also looked at the distribution of the age of tapes from last-use to the date-of-analysis, the current reserve-duty durations. The average site's library's median tape was already 78 days since last reference. Half the tapes in the library had been in reserve duty longer than 78 days. This latter set of statistics refute any idea that the high reference frequency within one week is a function of very short retention periods.

The cumulative frequency of the reserve-duty cycle, will later be used for evaluating various cache sizings.

Our other significant data source is the SMF type-21 error statistics by volume (ESV) record. There is one type-21 record corresponding to every tape dismount and thus corresponding to every mount. The fields which interest us are the VOLSER, date, and UCB-type (round-square shape status). The type-21 is irrelevant to DASD analysis and thus the analyst may have previously overlooked it. There are exactly the same number of type-21 records as the sum of Specific and Non-specific tape mounts in type-30 records for the corresponding intervals.

After sorting the type-21 data by shape and volser, we can count how many times each volume was mounted and recounted for a given period. Using SAS, we can create a Total Mounts variable, TMOUNTS, which carries the total number of mounts for each tape volume. PROC UNIVARIATE with the FREQ option can be subsequently unused to further characterize tape usage. Figure 3 shows the univariate output for one medium-size shop's cartridges for a one-week period.

The MEAN is the mean number of mounts per tape cartridge referenced in this week, (2.06477). The N is the total number of unique cartridge tapes (mounted in a week), 2146. The SUM statistic for this variable is the total number of mounts, 4431. Reading from the frequency distribution, 61.9 percent of the tapes were only mounted once during a week. However, this means

that tapes mounted only once constitute only 1329/4431 = 30% of the mounts. The other 70% of the mounts in this example come from the other 38% of the referenced tapes. The univariate output also indicates that the five most-popular tapes for this week were mounted between 24 and 41 times. Although the data was not as rigorously collected, it appears that no single volser ever made it to a shop's top-five list more than one day in a week's data. Clearly there is no such thing as a heavily accessed tape-volume over time. There is a significant shift to fresher tapes on a daily basis. This differs markedly from DASD where the same several volumes tend to have the heaviest usage every day for extended periods. Obviously we tend to update DASD datasets in place but we tend to write updated tape datasets to different tapes, probably at a higher generation number.

Before we leave the subject of data sources, I want to warn analysts to avoid using type SMF 14/15's (Dataset read and update) for inferring mounts. Although these records are essential for DASD analysis, it is unlikely that mount activity can be accurately determined. Reports based on 14/15's should be viewed with great skepticism. Amongst the 14/15's limitations, tapes may be mounted without a trace in a 14 or 15 if, the datasets is not opened (as in run-time JCL errors) a DD uses more than five volumes. A tape volume which has datasets which are opened and closed several times in a single step will not leave a clear mount history when viewed from the 14/15's. The type-21 record is easier to interpret and more accurate for our purposes.

DOES MOUNT-TIME MATTER?

How much do tape-mount delays matter? The question is complicated since no standard tape-mount time statistic exists in SMF or RMF. To answer the question we must first model how many mount-events a job must await. The mount-event is basically the number of unique operator interactions pertaining to a job. I define

$$M = T - D + 1$$

where:

M is the number of Mount-Events
T is the total number of tapes a job mounts and
D is the number of tape drives the job allocates.

I analayzed a set of 1701 jobs which used tapes, modelling the impact of 60 and 120 second average mount delays. If we assume a 60-second average mount delay per mount-event, then a mean of 32% of execution time is mount delay and 79% of the jobs had more mount-delay than CPU time. If we assume these same jobs were provided 120-second mount delays, then the mean proportion of execution-time due to mount-delay was 49% and 88% had more mount-delay than CPU time. (All CPU timings are based on 3084Q.) As CPU and cached DASD subsystems provide faster service, the proportions of execution time in mount delay will continue to increase rapidly.

We should also be aware that a tape-drive is fully captive to a job from device allocation-time until device deallocation. If one-third to one-half of the devices' allocated time is awaiting mount, reducing mount times substantially can provide a major increase in available tape drives. If we were to reduce the execution time of work by 25% mount-time improvements, this is equivalent to increasing the number of tape-drives by 25%.

What about the cartridge autoloader? Eighty-four percent of the tape jobs in our shop request at least one non-scratch tape. Since a job does not progress until all initial mounts are satisfied, autoloaders alone do not adequately meet the performance shortfall. Note, if any allocation awaits mount the whole job is stalled and the apparent tape-drive capacity increase will not accrue.

Mount time delay is the largest component in tape-based workloads' performance and its importance is growing.

STORAGETEK'S 4400 LIBRARY

The StorageTek 4400 is a blend of standard 3480 tape media and robotics. The literal core of this system is the Library Storage Module (LSM), a dodecahedragonal (twelve-sided) silo holding about fifty-five hundred 3480-compatible cartridges. About the circumference of the silo are StorageTek 3480-compatible tape drives and optionally additional LSMs. The average service-time to fetch and mount a resident cartridge is eleven seconds. The system works in cooperation with its Host Software Component (HSC) which intercepts all tape-mount requests and influences MVS to use a silo-attached or a manual drive depending on whether the 4400 can satisfy the request. Up to sixteen LSM's can be interconnected and tapes can be automatically passed between LSMs to satisfy mount requests. If the robot is incapacitated, the LSM can be opened for flesh and blood human operators to manually retrieve and mount cartridges.

The 4400 elicits one central question, what should it be used for? The 4400 is too expensive to use for storing a shop's entire cartridge library. One cannot devote the 4400 to the heavily-referenced tapes since this paper has already shown there are no heavily used tapes over an extended period.

TAPE LIBRARY CACHING

I propose that we exploit automated tape library systems such as the StorageTek 4400 system as a tape-library cache. By placing an adequate number of scratch tapes and recently-reference tapes within a 4400 and purging tapes unreferenced for some number of days, we can automate a very large portion of tape mounts. We also reserve a number of open cells for adding volumes which were not predicted (cache faults). Reserving vacant cells insures that we do not have to purge a tape as a precondition to adding a tape. Tape library caching will dramatically reduce the turnaround time of workloads which require tape-mounts and improve the consistency of their performance. The purging of unreferenced tapes and the replenishment of scratch tapes only takes place at scheduled Algorithm Service Intervals (ASI), not more than once per day. The basic scheme is illustrated in Figure 5.

To evaluate how well a tape-library cache would work, I built a simulator. The simulator expects that all a shop's scratch mounts are handled from the cache and that the shop removes tapes after some pre-determined number of consecutive days of inactivity, an Unreferenced Interval Count (UIC). The program simultaneously simulates the cache-hit behavior for UICs of 7, 14, and 28 days. (The simulator also allows one to select any three values one wishes to evaluate.) The three test-UIC limit-values are known as LIMIT_A, LIMIT_B, and LIMIT_C. An example of the simulation output constituted Figure 7.

Basically, the simulator uses the TMC to generate the contents of a cache with the prescribed UIC. It then applies the mount history from the type-21 data one day at a time. If a mount is made to a volume created within the selected LIMIT, all that volume's mounts are considered cache hits. If the tape was created earlier than the selected LIMIT-ago then we count the first mount for the volume as a cache miss and the rest as hits. In this case, however, we count all mounts to this volume in the subsequent several days' type-21 data, as cache hits. Total mounts for each day by shape are the tallied into cache hits and misses.

This is a very conservative simulator, due to the limitations of the input data. Because we use the early day's data to partically prime the simulated cache, the meaningful statistics come from the latter days' results.

By using the last full week-day's results, we get a very good indicator of how well the 4400 would function if it could house 7, 14, or 28 days' of tapes. Our earlier cumulative distribution of the reserve-cycle (Figure 6) tells us how many tapes are subsumed to support any particular UIC. By comparing the desired UIC population size to the vendor's offering, hardware sizing is trivial. For the STC LSM, we divide the 7-day reserve-duty cumulative frequency by 5500, the STC LSM capacity size. This yields the number of LSMs which are necessary to keep 7-days' tapes in the cache. Of course the simulator can be re-run specifying a smaller number of days to approximate smaller hardware configurations.

By using the TMC we automatically and accurately factor-out the ongoing conversion of tapes to scratch and elimination of tapes from the library (deletion). We can also remove off-site tapes from our cache population although this was not done in this preliminary research.

In actual operation we would pre-load the 4400 with the appropriate recently-referenced tapes. This would insure that we would achieve at least the simulated hit rates beginning on the first day of operation.

RESULTS

Nine sites participated in a study of tape-reference patterns and caching. Figure 4 summarizes the principle results of this research. I found that using a 7-day UIC, no shop achieved less than a 90% hit rate (mean = 95%). This proportion was achieved by placing a weighted average of only 17% of the shops' cartridge library into Library Storage modules. For example, Site 7 today performs 4,444 mounts in one typical day. The simulator found that using the LRU algorithm, 4,313 could be automated leaving only 131 manual mounts in one day.

If we assume that a STC Library Storage Module can accommodate 5500 tapes, four of the sites could store significantly more than seven days' tapes into a minimal (one-LSM) configuration. In these cases, better hit rates would be achieved than simulated by merely using all the hardware of a single LSM.

In all cases, there were very minimal gains and often very significant costs associated with maintaining 14 or 28-days' of tapes. I cannot recommend paying for additional capacity to reach deeply into this zone of diminishing returns.

Of course the 4400 must be fed and cared for at regular algorithm service intervals. Although the numbers of tapes involved in the ASI processing is large, we have successfully moved the manual aspect of tape processing from an operation which is synchronous to job-processing to one which is dominantly asynchronous to job processing. Furthermore, we can predict the impact on particular workloads whose reference patterns we know, such as those which only reference scratch and recently created tapes. For these workloads, tape mounting has been entirely automated.

CONCLUSION

Tape libraries are accessed with significant locality-of-reference which has not been previously reported. Because each tape mount is a commonly recorded discrete event (through the SMF type-21 record), the preferred method of performance predictions simulation rather than analytic modeling. Simulation allows a considered configuration to be tested without making any significant assumptions yielding highly robust results. All nine sites that were studied achieved better than 90% of their mounts automated by maintaining tapes in the system until they were unreferenced for seven consecutive days. This indicates that tape libraries are ideally suited for caching. Easy and inexpensive to run simulations can accurately identify anomalous environments, where caching maybe ineffective or cost-prohibitive. Using an automated tape library as a cache can make significant tape workloads run as consistently and quickly as current DASD-based workloads.

Figure 1

Library Tape Volume Duty Cycle

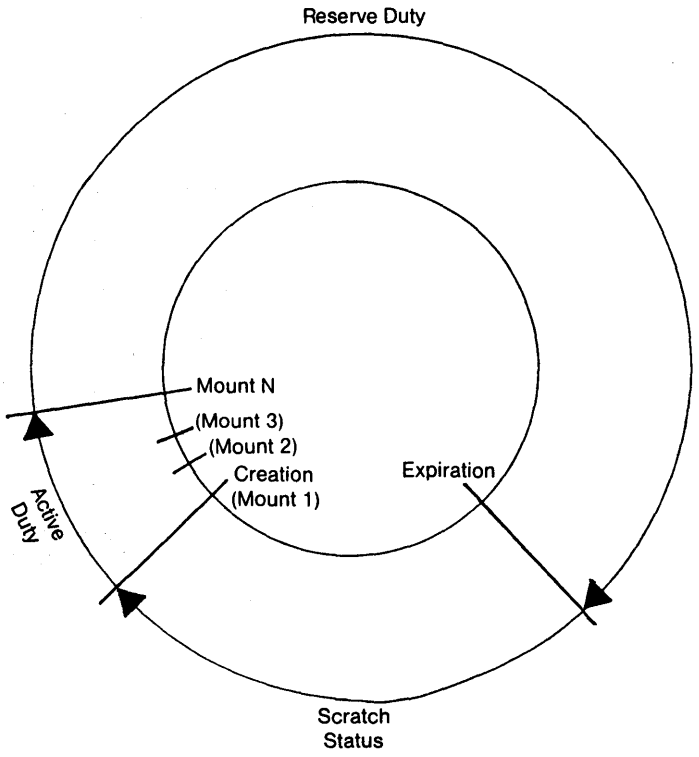


Figure 2

INTELLIGENT SILO MANAGER
IMPLEMENTATION DATA COLLECTION
STEP 2: DISTRIBUTION OF TAPES RECENTLY REFERENCED
SHAPE-INDEPENDENT

11:21 MONDAY, AUGUST 31, 1987 5

CUMULATIVE PERCENTAGE BAR CHART

DAZ2LST	DAYS FROM CREATION TO LAST REFERENCE	FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
UNDER 1	**********	24047	24047	68.38	68.38
1	**********	2193	26240	6.24	74.62
2	**********	618	26858	1.76	76.37
3	**********	430	27288	1.22	77.60
4	**********	350	27638	1.00	78.59
5	**********	281	27919	0.80	79.39
6	**********	251	28170	0.71	80.10
7	**********	302	28472	0.86	80.96
8	**********	209	28681	0.59	81.56
9	**********	167	28848	0.47	82.03
10	**********	125	28973	0.36	82.39
11	**********	115	29088	0.33	82.71
12	**********	110	29198	0.31	83.03
13	**********	137	29335	0.39	83.42
14	**********	196	29531	0.56	83.97
15	**********	159	29690	0.45	84.43
16	**********	86	29776	0.24	84.67
17	**********	130	29906	0.37	85.04
18	**********	112	30018	0.32	85.36
19	**********	104	30122	0.30	85.65
20	**********	101	30223	0.29	85.94
21	**********	105	30328	0.30	86.24
22	**********	78	30406	0.22	86.46
23	**********	96	30502	0.27	86.73
24	**********	105	30607	0.30	87.03
25	**********	49	30656	0.14	87.17
26	**********	69	30725	0.20	87.37
27	**********	103	30828	0.29	87.66
28	**********	77	30905	0.22	87.88
29	**********	129	31034	0.37	88.25
30	**********	101	31135	0.29	88.53
31	**********	93	31228	0.26	88.80
OVER 31	**********	3939	35167	11.20	100.00

10 20 30 40 50 60 70 80 90 100

CUMULATIVE PERCENTAGE

Figure 3

HEWITT ASSOCIATES INTELLIGENT SILO MANAGER
IMPLEMENTATION DATA COLLECTION
MOUNT ACTIVITY STATISTICS BY TAPE DRIVE TYPE

11:21 MONDAY, AUGUST 31, 1987

SHAPE=SQUARE

UNIVARIATE

VARIABLE=TMOUNTS

MOMENTS

N	2146	SUM WGTS	2146
MEAN	2.06477	SUM	4431
STD DEV	2.5595	VARIANCE	6.55105
SKEWNESS	6.36141	KURTOSIS	62.715
USS	23201	CSS	14052
CV	123.961	STD MEAN	0.055251
T:MEAN=0	37.3707	PROB>|T|	0.0001
SGN RANK	1151866	PROB>|S|	0.0001
NUM ^= 0	2146		

QUANTILES(DEF=4)

100% MAX	41	99%	12
75% Q3	2	95%	6
50% MED	1	90%	4
25% Q1	1	10%	1
0% MIN	1	5%	1
		1%	1
RANGE	40		
Q3-Q1	1		
MODE	1		

EXTREMES

LOWEST	ID	HIGHEST	ID
1	(702729)	24	(207820)
1	(702728)	25	(203986)
1	(702727)	28	(210899)
1	(702726)	37	(200544)
1	(702725)	41	(209375)

FREQUENCY TABLE

VALUE	COUNT	PERCENTS CELL	PERCENTS CUM	VALUE	COUNT	PERCENTS CELL	PERCENTS CUM	VALUE	COUNT	PERCENTS CELL	PERCENTS CUM	VALUE	COUNT	PERCENTS CELL	PERCENTS CUM
1	1329	61.9	61.9	7	22	1.0	97.3	13	1	0.0	99.1	23	2	0.1	99.7
2	366	17.1	79.0	8	10	0.5	97.7	14	2	0.1	99.2	24	2	0.1	99.8
3	180	8.4	87.4	9	10	0.5	98.2	15	2	0.1	99.3	25	1	0.0	99.9
4	108	5.0	92.4	10	7	0.3	98.5	16	1	0.0	99.3	28	1	0.0	99.9
5	43	2.0	94.4	11	5	0.2	98.7	17	4	0.2	99.5	37	1	0.0	100.0
6	39	1.8	96.2	12	7	0.3	99.1	18	2	0.1	99.6	41	1	0.0	100.0

THE N STAT IS THE TOTAL NUMBER OF TAPES IN THE ANALYSIS
THE SUM STAT IS THE TOTAL NUMBER OF MOUNTS TO THOSE TAPES
THE HIGHEST COLUMN (ON THE FAR RIGHT), SHOWS THE VOLSERS
MOST FREQUENTLY MOUNTED AND HOW MANY UNIQUE MOUNTS IN PERIOD

Figure 4
Summary of Nine Sites' Results

Company ID	#Square Tapes in Lib	Mounts in One day	7 days Refer'd	Sent Offsite	% of Lib	% Hits	#Auto-mated	#Manually Mounted	7-day Purge	14 days Refer'd	14 days % Hits	28 days Refer'd	28 days % Hits
1	8,338	440	2,682	287	32.2%	99.1%	436	4	235	3,612	99.3%	4,622	99.3%
2	29,927	3,511	8,158	1405	27.3%	98.6%	3,463	48	451	10,807	99.0%	13,799	99.3%
3	69,854	3,030	11,777	235	16.9%	92.3%	2,798	232	1,590	19,255	92.9%	28,456	93.9%
4	37,972	2,806	6,547	235	17.2%	91.3%	2,561	245	1,120	11,322	93.6%	16,944	94.9%
5	67,052	1,691	8,650	?	12.9%	94.0%	1,589	102	1,270	15,914	95.2%	23,541	95.6%
6	3,520	230	762	66	21.6%	99.6%	229	1	137	1,355	99.6%	2,138	99.6%
7	93,535	4,444	13,489	1272	14.4%	97.1%	4,313	131	1,364	22,099	98.3%	36,591	98.5%
8	12,906	560	1,843	362	14.3%	91.3%	511	49	209	3,178	92.0%	4,767	94.6%
9	5,299	375	2,035	208	38.4%	99.5%	373	2	277	3,074	100.0%	4,165	100.0%

Hewitt Associates Tape Reference Study

Company ID	Mean Tmounts	Most Freq'ly Mounted VOLSER/Day	Pct library Ref'd<7 days fr creation	Median Time Since Last Reference
1	1.34	18	89.9	28
2	1.64	23	87.0	75
3	1.46	14	79.1	79
4	1.61	.	82.5	35
5	1.54	.	67.3	116
6	1.27	6	94.0	24
7	1.55	115	85.8	105
8	1.39	13	81.2	210
9	1.32	9	95.0	32

INTELLIGENT LIBRARY MANAGER (ILM)

Figure 5

MARCH 8, 1987

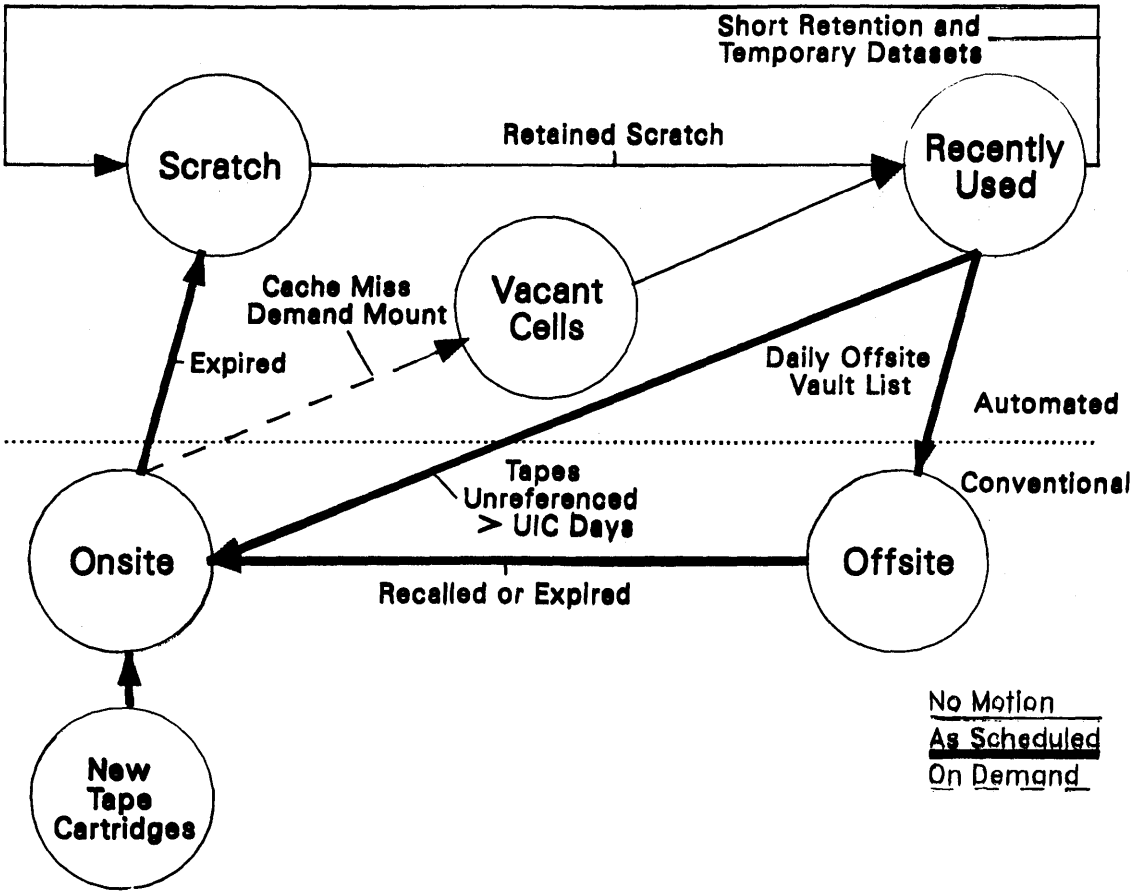


Figure 6

INTELLIGENT SILO MANAGER
IMPLEMENTATION DATA COLLECTION
STEP 2: DISTRIBUTION OF TAPES RECENTLY REFERENCED BY SHAPE

11:21 MONDAY, AUGUST 31, 1987 13

Reserve Cycle

SHAPE=SQUARE

LUSEDAGO	COUNT	PERCENT	CUMCOUNT	CUMPCT	LUSEDAGO	COUNT	PERCENT	CUMCOUNT	CUMPCT
UNDER 1	113	0.78	113	0.78	51	8	0.06	6,646	45.91
1	179	1.24	292	2.02	52	21	0.15	6,667	46.06
2	316	2.18	608	4.20	53	42	0.29	6,709	46.35
3	415	2.87	1,023	7.07	54	45	0.31	6,754	46.66
4	425	2.94	1,448	10.00	55	82	0.57	6,836	47.22
5	265	1.83	1,713	11.83	56	28	0.19	6,864	47.42
6	283	1.95	1,996	13.79	57	12	0.08	6,876	47.50
7	200	1.38	2,196	15.17	59	42	0.29	6,918	47.79
8	117	0.81	2,313	15.98	60	72	0.50	6,990	48.29
9	265	1.83	2,578	17.81	61	45	0.31	7,035	48.60
10	188	1.30	2,766	19.11	62	54	0.37	7,089	48.97
11	196	1.35	2,962	20.46	63	69	0.48	7,158	49.45
12	234	1.62	3,196	22.08	64	5	0.03	7,163	49.48
13	252	1.74	3,448	23.82	65	36	0.25	7,199	49.73
14	137	0.95	3,585	24.77	66	47	0.32	7,246	50.06
15	88	0.61	3,673	25.37	67	53	0.37	7,299	50.42
16	170	1.17	3,843	26.55	68	46	0.32	7,345	50.74
17	166	1.15	4,009	27.69	69	48	0.33	7,393	51.07
18	190	1.31	4,199	29.01	70	50	0.35	7,443	51.42
19	129	0.89	4,328	29.90	71	7	0.05	7,450	51.46
20	156	1.08	4,484	30.98	72	10	0.07	7,460	51.53
21	101	0.70	4,585	31.67	73	36	0.25	7,496	51.78
22	66	0.46	4,651	32.13	74	59	0.41	7,555	52.19
23	118	0.82	4,769	32.94	75	46	0.32	7,601	52.51
24	114	0.79	4,883	33.73	76	38	0.26	7,639	52.77
25	108	0.75	4,991	34.48	77	31	0.21	7,670	52.98
26	173	1.20	5,164	35.67	78	5	0.03	7,675	53.02
27	157	1.08	5,321	36.76	79	19	0.13	7,694	53.15
28	84	0.58	5,405	37.34	80	50	0.35	7,744	53.50
29	30	0.21	5,435	37.54	81	30	0.21	7,774	53.70
30	43	0.30	5,478	37.84	82	43	0.30	7,817	54.00
31	100	0.69	5,578	38.53	83	35	0.24	7,852	54.24
32	108	0.75	5,686	39.28	84	72	0.50	7,924	54.74
33	92	0.64	5,778	39.91	85	4	0.03	7,928	54.77
34	61	0.42	5,839	40.34	86	11	0.08	7,939	54.84
35	58	0.40	5,897	40.74	87	48	0.33	7,987	55.17
36	42	0.29	5,939	41.03	88	34	0.23	8,021	55.41
37	45	0.31	5,984	41.34	89	60	0.41	8,081	55.82
38	65	0.45	6,049	41.79	90	28	0.19	8,109	56.02
39	79	0.55	6,128	42.33	OVER 90	6367	43.98	14,476	100.00
40	50	0.35	6,178	42.68					
41	61	0.42	6,239	43.10					
42	52	0.36	6,291	43.46					
43	11	0.08	6,302	43.53					
44	8	0.06	6,310	43.59					
45	38	0.26	6,348	43.85					
46	84	0.58	6,432	44.43					
47	73	0.50	6,505	44.94					
48	74	0.51	6,579	45.45					
49	55	0.38	6,634	45.83					
50	4	0.03	6,638	45.86					

Figure 7

SUMMARY FOR SHAPE=SQUARE

MAX DAYS	#HITS(PCT HITS)	#MANUAL MTS(PCT MANUAL)
	THERE WERE 511 TOTAL MOUNTS ON FRIDAY, MAY 15, 1987	
7	302 (59.1%)	209 (40.9%)
14	374 (73.2%)	137 (26.8%)
28	410 (80.2%)	101 (19.8%)
	THERE WERE 292 TOTAL MOUNTS ON SATURDAY, MAY 16, 1987	
7	277 (94.9%)	15 (5.1%)
14	277 (94.9%)	15 (5.1%)
28	279 (95.5%)	13 (4.5%)
	THERE WERE 162 TOTAL MOUNTS ON SUNDAY, MAY 17, 1987	
7	159 (98.1%)	3 (1.9%)
14	160 (98.8%)	2 (1.2%)
28	162 (100%)	. (. %)
	THERE WERE 598 TOTAL MOUNTS ON MONDAY, MAY 18, 1987	
7	458 (76.6%)	140 (23.4%)
14	488 (81.6%)	110 (18.4%)
28	533 (89.1%)	65 (10.9%)
	THERE WERE 607 TOTAL MOUNTS ON TUESDAY, MAY 19, 1987	
7	503 (82.9%)	104 (17.1%)
14	532 (87.6%)	75 (12.4%)
28	558 (91.9%)	49 (8.1%)
	THERE WERE 618 TOTAL MOUNTS ON WEDNESDAY, MAY 20, 1987	
7	559 (90.5%)	59 (9.5%)
14	572 (92.6%)	46 (7.4%)
28	581 (94.0%)	37 (6.0%)
	THERE WERE 560 TOTAL MOUNTS ON THURSDAY, MAY 21, 1987	
7	511 (91.3%)	49 (8.8%)
14	515 (92.0%)	45 (8.0%)
28	530 (94.6%)	30 (5.4%)
	THERE WERE 1 TOTAL MOUNTS ON FRIDAY, MAY 22, 1987	
7	1 (100%)	. (. %)
14	1 (100%)	. (. %)
28	1 (100%)	. (. %)

NOTE: THE HIT RATIOS ARE ACTUALLY MINIMUM VALUES, ACTUAL VALUES WILL BE SIGNIFICANTLY BETTER. HIT-RATIOS FOR LATTER DAYS ARE MORE ACCURATE THAN EARLIER DAYS.

ANALYZING THE PERFORMANCE CHARACTERISTICS OF 18-TRACK TAPE TECHNOLOGY

William F. Dines, Jr. and Bradley A. Cahill
Storage Technology Corporation

ABSTRACT

With the announcement of 18-track cartridge technology in March of 1984, several new concepts were introduced which have set the direction for future tape products. Although all of the enhancements and characteristics of this new technology seem to indicate a significant performance advantage, in some cases there can be a decrease in tape throughput. A key component of this new architecture is the buffered control unit whose primary function is to mask the slow speed of its attached cartridge transports. This paper will discuss in detail the various features of 18-track buffered control units as well as those issues directly related to 18-track performance. Benchmark results will demonstrate the effect of such features as tape-write-immediate, high speed search and buffer management techniques (i.e. buffer stealing and buffer segment allocation) as well as the effect of blocksize on 18-track performance and capacity.

INTRODUCTION

As advances abound in the information storage and retrieval arena, the inherent performance benefits may not be as self-evident as they appear. One must go beyond the advertised or assumed advantages to assess the actual impact of the technology. A definitive example is the introduction of the 18-track cartridge tape technology. Data buffering and three megabyte per second data transfers are obviously major enhancements to tape performance, or are they? Unfortunately, too much emphasis is given to these benefits and not enough to technology changes such as "back-hitching" and "buffer stealing". Even something as simplistic as the use of small blocksizes has a negative impact on cartridge capacity and effective data rate throughput. The significance of blocksize is, in many cases, overlooked or not addressed because of the application changes required to support larger blocksizes.

It is the intent of this paper to go beyond the intuitive advantages of this relatively new tape technology and review, in detail, the salient characteristics of a new generation in magnetic tape development. The main focus will concentrate on the control unit portion of 18-track tape. The intent is to explore the full potential of all the features and facilities that are currently provided with this technology.

BACKGROUND

The requirement for high speed sequential tape processing is still an important factor in many data processing environments today. The use of 18-track tape has essentially remained the same as with older 3420 9-track tape technology: a growing requirement for online disk dataset and volume backups, for archiving aged and/or low usage disk files, for use as the primary data interchange facility, and for large sequential batch file applications which, for reasons of cost, cannot reside on disk.

The batch window, which utilizes tape resources the heaviest, is still primarily completed between the second and third shifts; that is, between the hours of approximately 7:00 P.M. to 7:00 A.M. As online availability requirements and workloads are expanded into these periods, several problems are encountered. First, the batch window is being compressed, putting a greater demand on the tape subsystem to complete the nightly workload in sufficient time to bring up the online system the following day. Second, any growth associated with online systems typically generates additional workloads to be supported by the batch window. "For a time, there is a classic rush hour effect with jobs waiting on tape resources. Gradually, jobs and networks complete and the backlog of tape demand is worked off". A further review of these tape processing requirements may be found in Friedman M. and Dines W. "Performance Measurement Of Buffered Tape" [1].

18-TRACK HARDWARE

The 18-track cartridge tape hardware can be segmented into three distinct components: media, tape transport, and control unit. A brief review of the media and the tape transport will be provided to illustrate the capacity and performance issues associated with these components. The control unit, which is the key element of the 18-track cartridge subsystem, will be analyzed in greater detail in the following sections.

The media consists of one-half inch Chromium Dioxide tape which is contained in a compact cartridge and is self-threading via a leader block. The dimensions of the cartridge are approximately 4" X 5" X 1". There are no beginning-of-tape (BOT) or end-of-tape (EOT) markers as there is on older 9-track 10.5 inch reel media. The cartridge drive automatically determines the beginning and end of tape.

The current Chromium Dioxide media supports a linear recording density of approximately thirty eight thousand bytes per inch (38Kb/inch). Data is recorded on eighteen

parallel recording tracks with an interblock gap (IBG) size of two millimeters or .08 inches between data records. Each cartridge contains a minimum of 541 feet of Chromium Dioxide tape.

When computing cartridge capacity, two other factors must be considered: data block overhead and unusable tape. There is approximately .009 inches of overhead associated with each data block written to tape and also, approximately 36 feet of tape that is unusable for data recording due to tape leaders, etc.. Because of these factors, 505 feet of usable tape and .089 inches of inter-block gap size are used for all calculations throughout this paper.

The significant features of the tape transport as they relate to the performance issues of 18-track technology are discussed in the remainder of this section. First, a nominal or instantaneous data rate of 3 megabytes per second contributes to performance. As stated previously, data is recorded on 18 parallel tracks with a linear byte recording density of 37,871 bytes per inch. The tape is moved on a reel to reel assembly at 79 inches per second (IPS) for normal read and write operations. For the high speed search facility, tape is moved at the rate of 157 IPS. This feature will be discussed in greater detail in subsequent sections.

Second, the transports are data streaming tape drives and require a significant amount of time to get up to speed, slow down and then backup to reposition the tape at the next record after an interruption. This is called a "back-hitch", and occurs whenever the continuous motion of the tape is interrupted. Because of this action, it is of the utmost importance to maintain a continuous read or write operation for as long as possible.

The third component of the 18-track cartridge technology is the control unit. This buffered, microprocessor driven control unit can connect up to 8 tape transports. Two control units can be configured via a dual control unit communications feature to support a total of 16 tape transports. The initial size of the buffer in a single control unit was 512Kb and has been recently increased to 1 megabyte. Each buffer is divided into 64Kb segments. These buffer segments are allocated to tape drives dynamically and are managed with a least recently used (LRU) algorithm based on tape drive activity. In a two control unit configuration, the subsystem will attempt to balance the workload in order to prevent an overload condition on one control unit. An overloaded control unit can transfer the control of a tape drive to its companion control unit. When this transfer is accomplished, the original buffer space is released and new buffer space is allocated in the other control unit. This is done to optimize the use of the buffer space across both control units and to balance internal path utilization.

The allocation of the buffer segments to a drive can be from zero to four 64Kb segments. For example, a fully configured tape system with 16 tape transports could have 2 segments allocated to each of the 16 drives ((2 * 16) * 64Kb)) = 2048Kb or 2 megabytes of available buffer space. If there is no activity to a tape drive, then the buffer space allocation for that drive would be zero and any active tape drives in the subsystem would be able to use the idle drives buffer segment.

The data buffer is the most critical resource in the 18-track tape subsystem. Data movement between a tape drive and the buffer may be done concurrently with data movement between the channel and the buffer. Data may be written to a buffer segment while that same data is being transferred from the buffer to the tape drive. Once again, the purpose of the buffer is to mask the relativly slow speed of the tape drive from the channel.

There are two distinct types of buffered operations: read and write. Both of these operations are supported differently. A buffered write operation is very straightforward. If the data path is available, data is transferred at channel speed from the channel into the data buffer segment and a "channel end" and "device end" is returned to the host, indicating that the write operation has been completed. The channel is then disconnected from the control unit and may service other operations on the path. It is the responsibility of the control unit on a buffered write operation to ensure data integrity, since the data has not yet been written to the tape drive at this point. Once the data begins to enter the buffer on a write operation, the control unit may begin data transfer from the buffer to the tape drive. If a subsequent data block has been transferred into the buffer for that same tape drive before the previous data block has completed its write operation, then the tape motion can be maintained and the buffer will continue to write the next data block without any interruption in tape movement (i.e. data streaming mode). There are two exceptions to this process: "Tape-Write-Immediate" and "Device Syncronization". These features are reviewed in a subsequent section.

On a buffered read operation, if the data is in the buffer, then it is transferred immediately to the host. If there is no data in the buffer when the host requests it, then the control unit defers the request and disconnects from the channel. The control unit then brings the requested tape drive up to speed and starts transferring data into the buffer. While data is being placed into the buffer, the control unit will attempt to reconnect to the channel in order to begin data transfer to the host. Under microprocessor control, the control unit performs read ahead functions in order to have the requested data in the buffer when needed so data transfer could take place immediately at channel speeds without any elongated start up time or delays.

There is an event that occurs which will cause the subsystem to read the same data blocks from a tape transport into the buffer more than one time. This event is known as buffer stealing and occurs when infrequently referenced buffer segments are reallocated to service more active tape transports. The tape transport that has had it's buffer area stolen is instructed to back up and reposition the media to a specific BLOCKID and wait for the next read request from the control unit.

Buffer stealing usually occurs when more than 50% of a maximum configured subsystem is active and it occurred more often when the buffer size was only 512Kb.

THE BENCHMARK PROGRAM

Throughout this paper, 18-track performance numbers are discussed. Many of the reported numbers were generated by a program written specifically to exercise various 18-track features and produce statistics on that activity. This program is available upon request.

18-TRACK CARTRIDGE CHARACTERISTICS AND CAPACITY

Data blocks and tape marks that are written to 18-track media are each assigned a physical "BLOCKID" number by the control unit. These BLOCKID's are imbedded within the data on the media and are not accessible through normal read operations. Groups of many BLOCKID's are assigned to larger areas on the media called "physical reference values" or "sectors". There are approximately 90 of these sectors on each cartridge. Analogous to DASD, one should think of 18-track technology sectors as DASD cylinders and BLOCKIDs as DASD tracks. BLOCKID's and sectors are referenced throughout this paper.

One of the most important, but often overlooked, characteristics of the 18-track cartridge is the inter-block gap (IBG) size (the space that lies between each data block on the media). The IBG size of current 18-track technology is .08 inches wide. One may look at past tape technologies and assume that this smaller IBG size is just one more advantage, but look again. It is not the size of the IBG that is important, but the amount of tape capacity that is wasted each time an IBG is created. For example, 1600 BPI 9-track technology has an IBG size of .6 inches and a linear recording density of 1600 bits per inch. This calculates to 960 bytes (1600 X .6) of wasted media capacity each time an IBG is created. With the introduction of 6250 BPI 9-track technology, the IBG decreased to .3 inches but the linear recording density increased to 6,250 bits per inch. Although the size of the IBG decreased, the amount of data lost by the creation of an IBG increased 95% to 1,875 bytes. Fortunately, this increase in wasted capacity due to the IBG is offset by the fact that 9-track tape reels remained at 2400 feet and the recording density did indeed increase almost three-fold.

Unfortunately, this is not the case with 18-track cartridges. As mentioned earlier, the IBG size for 18-track cartridges is .08 inches wide and the recording density is 37,871 bytes per inch. It was also mentioned that there is an additional .009 inches of overhead associated with each data block, for a total IBG/overhead size of .089 inches. A quick calculation shows us that there is 3,370 bytes of cartridge capacity wasted each time that a data block is created. Think for a moment about all of the tape files that have a blocksize less than 3,370 bytes. That's right! The gaps are bigger than the data blocks, and more than 50% of a full cartridge is blank tape. These 3,370 bytes become even more important when one realizes that the amount of tape that is available decreased from 2,400 feet on 9-track to 541 feet on 18-track.

The role of blocksize is indeed a key issue. Obviously, the larger the blocksize, the fewer the number of IBG's. The fewer the number of IBG's, the lower the amount of wasted catridge storage capacity. Figures 1 and 2 illustrate this point and compare the capacity of a cartridge when the data is blocked at 2,000 bytes to the capacity of a cartridge when the data is blocked at 32,000 bytes. Notice that the cartridge's capacity increased 143% just by reblocking the data.

BLOCKSIZES AND CARTRIDGE TAPE CAPACITY

TAPE DENSITY --- 37871 BPI (18-TRACK)
BLOCKSIZE --- 2000 BYTES

$$\frac{2000}{37871} = .053 \text{ IN.} + .089 \text{ IN.(GAP\&OVHD)} = .142 \text{ IN.}$$

$$505 \text{ FT.} \times 12 = \frac{6060 \text{ INCHES}}{.142 \text{ INCHES}} = 42{,}676 \text{ REC.(85MB)}$$

REBLOCK FROM 2000 BYTES TO 32000 BYTES

BLOCKSIZE --- 32000 BYTES

$$\frac{32000}{37871} = .845 \text{ IN.} + .089 \text{ IN.(GAP\&OVHD)} = .934 \text{ IN.}$$

$$505 \text{ FT.} \times 12 = \frac{6060 \text{ INCHES}}{.934 \text{ INCHES}} = 6{,}488 \text{ REC.(207MB)}$$

Figure 1.

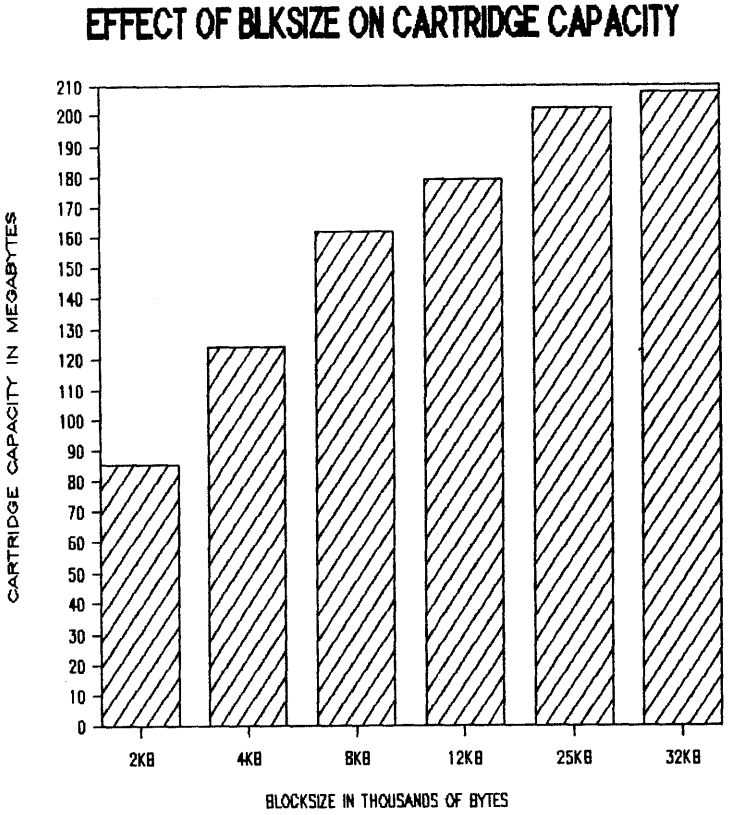


FIGURE 2.

There are other advantages to reblocking files to 32K even if the file is less than a full cartridge. Figure 3 will illustrate this point. If one processes a full cartridge blocked at 2,000 bytes, it takes 42,676 I/O's to access 85.35 MB of data. If that same file is reblocked to 32,000 bytes, only 6,488 I/O's are required to access 207.62 MB of data. That is, you can transfer 2.4 times as much data with 6.6 times less I/O's. Those advantages obviously relate to improvements in performance. Users with extremely small files on 18-track cartridges may consider moving those data sets to high capacity DASD.

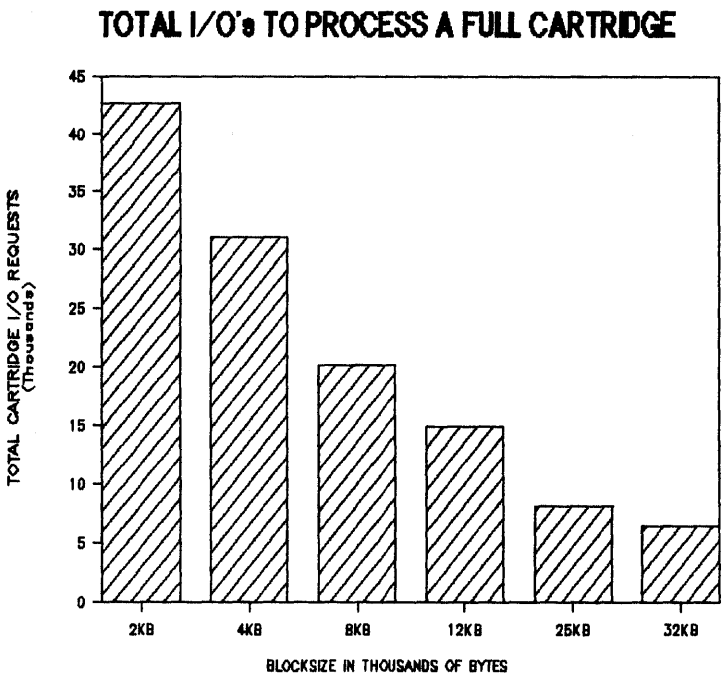


FIGURE 3.

18-TRACK TECHNOLOGY HIGH SPEED SEARCH

High speed search, a new feature with full function mode 18-track tape technology, allows users to randomly access sequential cartridge files. The high speed search feature uses the enhanced BSAM NOTE and POINT macros and/or the enhanced OPEN/CLOSE support to execute two new channel commands. These channel commands process the file BLOCKIDs mentioned in the section entitled 18-TRACK TECHNOLOGY CARTRIDGE CHARACTERISTICS AND CAPACITY. The first channel command associated with high speed search is the NOTE macro "READ BLOCK ID" command. This command transfers two 4 byte data blocks from the 18-track control unit to the host. These two data blocks describe BLOCKID's of records currently being processed or next to be processed by the subsystem. Should a user ever want to perform a high speed search to a particular BLOCKID, it is necessary to "note" the BLOCKID for the desired block and store it somewhere so that it can be retrieved at a later time and used in a "point" function. The second channel command associated with high speed search is the POINT macro "LOCATE BLOCK ID" command. This command transfers one four byte data block from the host to the control unit which describes the BLOCKID of the data block to be located. Information regarding these new channel commands was obtained from the manuals that are referenced in the bibliography.

So what is wrong with high speed search? Users should ask themselves this exact question if they plan to use it for random reads to 18-track files. Prior to high speed search, a user processed a typical tape file by opening it and reading it from start to finish using the QSAM access method. QSAM, as most users know today, is a very efficient access method and will process a sequential file much faster than its predecessor, BSAM. So what's the first thing that has to be done to use high speed search? The application has to be converted from QSAM back to BSAM. This requirement brings up a very interesting question. Is high speed search with BSAM faster than full file reads using QSAM? Well, it depends on how far into the file the high speed search is going and how many records are processed by BSAM once the desired record is found.

In Figure 4, two jobs were executed to demonstrate that, in some cases, high speed search using BSAM can be slower than reading an entire file using QSAM. The first job opened an 18-track file and read the complete file (4,000 BLOCKS) using QSAM. The execution time for this job was 30.669.7 seconds. The second job opened an 18-track file, performed a high speed search to decimal BLOCKID 02000150 (bypassed 146 data blocks) and read the remaining 3,854 data blocks using BSAM. The execution time for the second BSAM job was 37.860.8 seconds. The BSAM job using high speed search ran 23% longer than the QSAM job.

18-TRACK PERFORMANCE STATISTICS

HIGH SPEED SEARCH

QSAM FULL FILE READ VS. BSAM HIGH SPEED SEARCH

HIGH SPEED SEARCH WAS TO BLOCKID 02000150(DEC)

FILE BLOCKSIZE IS 16K BYTES

I/O WAIT IN MICROSECONDS

ALL TIME VALUES ARE IN FORMAT -- MM.SS.MLS.MCS

	QSAM READ ALL REC	BSAM WITH HSS
I/O COUNT	4,000	3,854
DELAY TIME	00.00.000.080	00.00.000.080
EXEC TIME	00.30.699.700	00.37.860.800

FIGURE 4.

The degradation in performance that can occur due to high speed search with BSAM can also impact the performance of the entire subsystem. In order to explain the additional contributing factors, we must first look at high speed search or the POINT macro instruction in more detail.

A POINT macro is issued by an application program when the program requires access to a record located some distance in front of or behind the read/write head. The format of the point macro instruction is:

```
[ symbol ]  POINT     dcbaddress
                       ,blkaddr
                     [,TYPE=ABS|REL]
```

dcbaddress

address - ADDRESS OF A DCB IN A-TYPE ADDRESS OR (2-12).

blkaddr

blkaddr - SPECIFY ADDRESS OF PHYSICAL BLOCK ID OR OF THE RELATIVE POSITION NUMBER.

TYPE=

ABS|REL - SPECIFY TYPE OF BLOCKID POINT REQUEST.
- ABS - PHYSICAL BLOCKID IS SUPPLIED IN BLKADDR.
- REL- SUPPORTS TRADITIONAL WAY OF SEARCHING TAPE.

EXAMPLE - SEARCH TO BLOCKID "07001000"

```
GOPOINT   POINT INPUT,BLOCKID,TYPE=ABS
.....     ...   .........
.....     ...   .........
INPUT     DCB   DDNAME=TAPE,DSORG=PS,MACRF=R,
                EODAD=X
.....     ...   .........
BLOCKID   DC    XL4'07001000'
```

In this example, let's assume that the cartridge tape file has been opened and the 18-track read/write head is currently positioned at BLOCKID X'01000004' (sector 01 and block 0004). The application wants to begin reading the file at BLOCKID X'07001000' (sector 07 and block 4096). Under these circumstances, the following are some of the events that would take place.

1. The POINT macro is issued.

2. The 18-track Control Unit receives the "LOCATE BLOCK" command along with the BLOCKID and then disconnects from the channel. The channel is then free to do other work.

3. The 18-track Control Unit then instructs the drive to begin forward motion at a speed of 157 inches per second and to reconnect when the drive reaches the area that is 2 sectors (sector 05) before sector 07. The tape transport then disconnects from the control unit and begins its high speed search. While the drive is disconnected from the control unit, the control unit is able to service another transport. When the high speed search transport reaches sector 05 (2 sectors before the target), it reconnects to the 18-track control unit and slows down to a search speed of 79 inches per second. At this time the control unit is dedicated to servicing the high speed search transport only.

4. The 18-track control unit and transport then perform a "slow speed search" until the BLOCKID X'07001000' is found. When the BLOCKID is found, the 18-track control unit signals end of operation and the next channel command is issued.

The most interesting portion of this high speed search is the fact that once "slow speed search" begins, the 18-track control unit is dedicated to servicing one transport until the desired BLOCKID is found. This type of "control unit lockout" can take as long as several seconds to complete, and this delay degrades the performance of the other transports in the subsystem.

In order to demonstrate the amount of delay time that exists when crossing physical reference areas, it is necessary to execute a job that performs a high speed search to the first BLOCKID in a sector of file and then performs a high speed search to the last BLOCKID in that same sector. The difference between the two time values is the time that it takes to cross the full sector. The results of these test are reported in Figure 5.

18-TRACK PERFORMANCE STATISTICS

ALL TIME VALUES ARE IN FORMAT -- MM.SS.MLS.MCS

BLOCKID NUMBERS ARE IN DECIMAL

	HIGH SPEED SEARCH TO '08001668'	HIGH SPEED SEARCH TO '08001936'
I/O COUNT	1	1
BLOCKSIZE	16,000	16,000
EXEC TIME	00.08.614.800	00.10.435.000

APPROX TIME TO CROSS SECTOR 08 IS 1.8 SECONDS

Figure 5.

The results of this test can be calculated. For example, the first and last BLOCKID's are known, so the number of total BLOCKID's in SECTOR 08 is easily calculated (See Figure 6). The number of BLOCKID's in SECTOR 08 is 269.

```
    TAPE DENSITY    --- 37871 BPI ( 18-TRACK )
    BLOCKSIZE       --- 16000 BYTES

16000
------ = .422 IN. + .089 IN.(GAP&OVHD)=.511 IN.
37871

       PHYSICAL REFERENCE AREA VALUE 008

    FIRST BLOCKID = 08001668 (decimal number)
    LAST  BLOCKID = 08001936 (decimal number)
     TOTAL BLOCKS =      269 (decimal number)

   269 BLOCKS X .511 INCHES = 137.46 INCHES
   (*   LENGTH OF PHYSICAL REF AREA 08   *)

    137.46 INCHES
  ----------------  = 1.74 SECONDS
       79 IPS
```

Figure 6.

If the blocksize is 16,000 bytes, then each block uses approximately .422 inches of tape. If the IBG/overhead length of .089 inches is added to the block length, then it is known that each block actually uses about .511 inches of tape. Multiply .511 inches times 269 blocks and the result is 137.46 inches. Divide the length of the sector by the speed of the transport (79 IPS), and the quotient is the amount of time that it takes to cross the sector. That calculated time value is 1.74 seconds, which is very close to what is reported by the benchmark program.

Remember, the benchmark program only reports the amount of time to cross one sector. It was stated that the slow speed search begins at 2 sectors before the target, which means that if each sector is approximately the same length, then the control unit lock-out on the drive port adapter would be around 5 seconds to locate BLOCKID 08001936. A high volume of this type of activity can severely degrade 18-track performance.

If high speed search is being used, and the blocksize of the blocks being searched is small, single microprocessor 18-track control units may be too busy to service the channel adapters requests, thus locking out data transfers to the host. Dual microprocessor 18-track control units do not have this problem because one microprocessor is dedicated to servicing the channel adapter and one is dedicated to servicing the drive port adapter.

TAPE WRITE IMMEDIATE

Tape-write-immediate (TWI) or write-validity-check is a special mode of operation for write events to 18-track transports. This mode of operation can be requested by coding OPTCD=W in the DCB parameter. It differs from normal write buffered mode in that the 18-track control unit does not signal device end to the host until the data block has been physically placed on the media and verified. This special mode of operation is necessary to service those applications that require a data block to be permanently stored on the media before continuing with any subsequent processing. Because of this type of I/O requirement, tape-write-immediate generally applies to BSAM, EXCP or QSAM (QSAM with BUFNO=1 specified only) applications and applications that code RECFM=F in their DCB and/or do their own data block management.

In normal buffered write operations, data is sent from the host to the control unit. The control unit then places the data into a buffer segment and responds channel end and device end to the host, which allows the requesting application to be dispatched. At some time the transport is brought up to speed, and the data is then physically placed on the media. This I/O overlapping type of operation is what makes the 18-track transport appear to be very fast when indeed it is slower than any of the more recent 9-track tape products.

When a user requests tape-write-immediate, the above mentioned I/O overlapping activity is not allowed to occur. Instead, the data is sent from the host to the control unit. The control unit, while placing the data into a buffer segment, requests that the transport be brought up to speed and then the data is moved from the buffer to the media. When the last byte has been written to the media and verified, the control unit signals end of operation to the host. The host waits for the transport to come to a stop and then backs up to position the media for the next start up operation before it can initiate the next I/O request. This repositioning action of the transport is called a back-hitch, and it can contribute 90 milliseconds of access delay time to each I/O. To put this into perspective, most 9-track tape transports have an access delay time of only 1.5 milliseconds.

So should tape-write-immediate be used? Tape-write-immediate can be used if the application's I/O requirement to 18-track tape does not exceed 10 I/O's per second. In other words, if the application has 90-100 milliseconds of processing time between each tape I/O then this time will allow the 18-track transport to reposition itself, thus masking its motion delay.

If a particular application has to use tape-write-immediate, then there are certain tuning efforts that can be invoked to reduce the impact of this feature. For example, let's assume that a user has previously used 9-track tape for an application and is forced to use 18-track tape with tape-write-immediate. The first option is to reblock the file (if possible) to a greater blocksize. This type of action has three advantages. First, by bringing in more data per I/O, the user decreases the number of I/O operations and thus reduces the number of back-hitch operations and their associated delays. Second, the reduction in I/O operations means a savings in the time involved in MVS IOS overhead. And third, the increase in the amount of data brought in with each I/O would increase the amount of time required to process the entire block. This increase in processing time may help to mask the delay time due to back-hitching.

To illustrate the effect that blocksize has on tape-write-immediate, two batch jobs were executed. One of the batch jobs performed normal buffered write operations at various blocksizes and the other performed tape-write-immediate write operations at various blocksizes. Figure 7 reports the results of the test. Notice that the BSAM job using normal buffered write operations starts off using 2,000 byte blocks with 10 milliseconds of processing delay time for each block. The execution time for this job is approximately 7 minutes and 31 seconds. The BSAM job using tape-write-immediate write operations also starts off using 2,000 byte blocks with 10 milliseconds of processing delay time for each block but runs approximately 53 minutes and 58 seconds. The tape-write-immediate job ran 7 times longer. The 10 milliseconds of processing delay time is not enough to mask the 90 millisecond back-hitch delay time of the 18-track transport.

It is assumed that if it takes 10 milliseconds to process the 2,000 byte block, then it takes 160 milliseconds to process the 32,000 byte block. By reblocking the data, three things

happen. First, both jobs perform 30,000 fewer I/Os, which means 30,000 fewer trips through IOS. Second, there are 30,000 fewer back-hitching events. And third, the increase in processing time for each block gives the 18-track transport enough time to back-hitch; thus, the tape-write-immediate job has a means of masking some of the motion delay of the transport.

So what happens to performance? By using 32,000 byte blocks, the execution time for the normal BSAM job is reduced to approximately 5 minutes and 48 seconds. The big improvement, however, is in the job that used tape-write-immediate. The execution time for this job is reduced to approximately 7 minutes and 13 seconds (See Figure 7) bringing it closer to the execution time of the job that did not use TWI.

BSAM TAPE WRITE IMMEDIATE

I/O DELAY BETWEEN I/Os IS IN MILLISECONDS

IN ALL CASES, 64 MEGABYTES WAS TRANSFERRED

ALL TIME VALUES ARE IN FORMAT -- MM.SS.MLS.MCS

	BSAM	BSAM-TWI
2K - I/O COUNT	32,000	32,000
8K - I/O COUNT	8,000	8,000
16K - I/O COUNT	4,000	4,000
25K - I/O COUNT	2,560	2,560
32K - I/O COUNT	2,000	2,000
2K - I/O DELAY	00.00.010.000	00.00.010.000
8K - I/O DELAY	00.00.040.000	00.00.040.000
16K - I/O DELAY	00.00.080.000	00.00.080.000
25K - I/O DELAY	00.00.125.000	00.00.125.000
32K - I/O DELAY	00.00.160.000	00.00.160.000
2K - EXEC TIME	07.31.542.400	53.58.403.100
8K - EXEC TIME	06.08.095.400	13.51.109.300
16K - EXEC TIME	05.55.851.300	08.46.614.700
25K - EXEC TIME	05.51.074.600	07.38.224.300
32K - EXEC TIME	05.47.725.000	07.13.132.100
2K - EDR	141,000	19,000
8K - EDR	173,000	77,000
16K - EDR	179,000	121,000
25K - EDR	182,000	139,000
32K - EDR	184,000	147,000

(EDR = EFFECTIVE DATA RATE)

Figure 7.

SYNCDEV MACRO

There are other methods besides tape-write-immediate that can be used to ensure that data is stored on the media before another event occurs. There is a new macro called "SYNCDEV". This macro controls data synchronization for 18-track devices that support full function buffered write and relates to write operations only. The SYNCDEV macro instructs the 18-track control unit to write any buffered data in the control unit to the addressed transport before another event can occur for that transport. The SYNCDEV macro is only 100% effective at the logical record level when RECFM=F is specified in the DCB, and either BSAM or EXCP is used or QSAM is used with DCB option BUFNO=1 in effect. Information regarding the SYNCDEV macro was obtained from the manuals referenced in the bibliography.

SUMMARY AND FUTURE CONSIDERATIONS

When used properly, 18-track technology is indeed a high capacity, high performance device. Even first generation 18-track products have inherent advantages over older unbuffered tape technologies because they fully utilize current channel speeds. As demonstrated, something as simple as blocksize plays an important role in both performance and capacity. Establishing large blocksizes as the standard should be a goal for all users of 18-track technology. Making cost effective use of each cartridge should also be a goal. Users may consider stacking smaller tape data sets on one cartridge or moving them to high capacity DASD.

It is also important for users to fully understand the performance implications associated with such features as tape-write-immediate and high speed search. As this paper has demonstrated, tape-write-immediate can severely degrade the performance of an application and the term "high speed search" can, in some cases, be deceiving. Although available as a feature, high speed search most likely is not intended to be used by applications to do random reads to sequential files. It is, however, very valuable for certain functions such as storage management products and error recovery.

As for the future, users can expect nothing but enhancements. Larger buffer sizes and more sophisticated buffer management algorithms are already available and have superceded the initial 18-track offering. One can expect increased linear recording densities, as well as a doubling or tripling of the current number of recording tracks available on current technology. These enhancements necessitate the requirement of faster channels (i.e. 4.5 or 6.0 megabytes per second) and would also provide users with increased cartridge capacity.

And finally, users can look foward to increased automation of 18-track technology. The inherent characteristics of the cartrige size and the way that tape can now be handled has invited the use of automation.

BIBLIOGRAPHY

[1] Friedman M. and Dines W., "Performance Measurement Of Buffered Tape", CMG XV Proceedings, 360-365 1984

IBM 3480 Magnetic Tape Subsystem Reference: Channel Commands, Status and Sense Bytes, and Error Recovery Procedures.

IBM 3480 Magnetic Tape Subsystem User's Reference.

7th Draft - Proposed American National Standard Recorded Magnetic Tape and Cartridge For Information Interchange - 18-Track.

STATIC VS DYNAMIC MANAGEMENT OF CONSISTENTLY VERY ACTIVE DATA SETS

Martha R. Henley and Ingrid Liu
International Business Machines Corporation
San Jose, CA

ABSTRACT

This paper describes the results of analyzing data from 12 MVS complexes. The data sets that were consistently very active during prime shift were identified. These data sets were analyzed to determine how much memory would be required if they were statically placed in an electronic store. In addition the fraction of total non-paging I/O on the system due to these specially managed data sets was determined. These data sets were also analyzed for locality of reference; a hit ratio was determined as a function of the size of the electronic store that was being dynamically managed. All of the consistently very active data sets had good enough locality of reference that a dramatic advantage was seen in managing this data dynamically rather than statically.

The analysis was applied to the problem of evaluating the effectiveness of a cache control unit vs a solid state device. Practical assumptions were made about memory sizes of the solid state device and their prices. Similarly assumptions were made about prices for the cache control unit and what volumes would be attached. For an equal cost cache and solid state device, the cache control unit handled about 1.7 times more I/O than was handled by the solid state device.

The results shown in this document are based on data collected from twelve installations. Individual installations may differ from those considered in this study.

Introduction

Purpose

Typically, a small number of very active data sets represents a large portion of the I/O activity of MVS systems. It is important to understand the characteristics of these very active data sets; overall I/O subsystem performance can be improved through proper management of these data sets. The purpose of this study was to:

1. Identify the consistently very active data sets in several customer installations.
2. Identify the characteristics of these data sets.
3. Evaluate the benefits of two different methods of managing these data sets in an electronic store.
   a. Static Management, or simply loading data into the store and leaving it there.
   b. Dynamic management, or bringing the data into the store as it is referenced and bumping it out of the store on a least recently used basis.

General Approach

In the summer of 1985 seven customers were asked to collect data from a total of 12 data processing complexes. RMF data, dumps of all VTOCs, and GTF SIO traces for all devices were collected. The trace data was taken for two 15-minute periods, one a morning period, and one an afternoon period. The trace times were chosen by the customer to represent typical heavy load times. The traces were timed so that they coincided with the RMF intervals, and were taken simultaneously for all systems in a JES complex. General information about the installation such as configuration and overall workload was obtained from the RMF data. The very active data sets of each customer were identified using VTOC dumps and GTF SIO traces.

Static management of the consistently very active data sets was evaluated by determining what fraction of the non-paging I/O on the system would be to the data sets in the electronic store. The dynamic management was evaluated using Cache Analysis Aid (CAA). The dynamic management algorithms assumed in this simulation program matched those used in the 3880 Model 23 control unit. In other words, the data was moved into the cache when it was referenced starting at the point of reference and continuing to the end of track; the data was removed from the cache on a least recently used basis. Note that although the simulation program used was oriented toward a cached control unit, the hit ratio predictions can be generalized to a main memory buffer managed on a least recently used basis.

The remainder of this report is divided into four sections plus a conclusion. The four sections address:

1. A summary of the workload characteristics of the installations that were studied.
2. The characteristics of the consistently very active data sets in each installation and their classification into four categories.
3. The behavior of the four categories of consistently very active data sets and the amount of I/O captured depending on the category, the size of the electronic store, and the management choice.
4. An analysis of the results a customer might obtain from a cache control unit and a solid state device.

Customer Workload Characteristics

In order to put the detailed results of the study into perspective, it is useful to have some idea of the size of the installation, the type of equipment installed, and the nature of the applications running on the system. This section shows the results of processing the RMF data for the particular interval in the morning for which SIO trace data was available. Thus, it reflects the workload level and mix during the time that data set activity level could be identified. Figure 1 shows the processor models, processor utilization, number of DASD volumes by model, the DASD I/O rates, GB/MIPS, and KIPIO for each of the complexes studied. The gigabytes (GB) on-line were calculated from the device capacities of the on-line DASD volumes. The millions of instructions per second (MIPS) were ballpark estimates based on the CPU model. The thousands of instructions per DASD I/O (KIPIO) was calculated using the MIPS, CPU busy, and the DASD I/O rate. The customers are identified by the first digit of the complex number, and the installation within customer is given in the second digit. There were seven customers with a total of twelve complexes studied. The workload characteristics are summarized as follows:

Characteristics by complex

Cmplx ID	CPU Model	CPU Util	DASD Type	DASD Vols	DASD IO/Sec	GB/ MIPS	KIPIO
11	3081K 3033	98.6% 96.9%	3380 3350 3330 2305	128 22 26 3	305.2 49.3 70.2 19.3	4.8	39.6
21	3084P 3084P	60.5% 64.5%	3380	168	403.9	3.7	43.2
22	3033 3084P 3084P	70.1% 76.0% 77.5%	3380 3350	272 2	673.1 0.0	5.1	39.6
31	3083 3081GX	68.1% 54.5%	3380 3350	68 109	153.9 223.6	3.9	30.0
41	3081G 3081D 3084Q	99.9% 73.8% 61.2%	3380 3350	249 40	510.3 71.0	3.5	56.1
51	3084P 3084P 3084Q	77.7% 77.1% 65.3%	3380 3350	232 76	726.7 110.9	3.1	45.2
52	3084P 3084Q	73.6% 66.9%	3380 3350	208 47	629.7 33.9	3.5	42.2
53	3081G 3084P	53.4% 33.9%	3380	132	223.9	3.2	46.2
61	3081K 3081K 3084Q	95.2% 81.1% 78.3%	3380 3350	336 442	492.1 481.2	6.3	47.4
62	3081K 3084Q	64.5% 99.0%	3380 3350 3330 2305	231 448 12 4	277.7 453.5 13.4 10.4	7.3	46.1
63	3081K 3081K	65.3% 52.3%	3380 3350 3330 2305	256 483 4 2	216.0 285.7 3.0 4.9	11.0	32.3
71	3081K 5860 3084Q	65.9% 99.9% 80.6%	3380E 3380	47 335	103.3 292.3	5.0	107.2

Figure 1. Customer Characteristics

Almost all of the customers studied were using MVS/XA. Two had complexes where there was a mix of MVS/SP and MVS/XA systems. About half the complexes were running with JES3 and about half with JES2. The total MIPS estimated for each of the complexes ranged from about 19 to about 56. These are considered to be medium-to-large installations. Most of the complexes were predominantly 3380, although two customers still had a significant portion of their DASD capacity in 3350 devices. Only one customer had installed 3380-E boxes. This is not surprising since the data was collected soon after the announcement of the larger capacity devices.

Fraction of CPU Utilization and EXCPs by Workload Type

The RMF workload activity records were processed to determine the fraction of the load on the system due to various application systems. The Installation Performance Specification from each customer was used to deduce what performance groups and/or domains were associated with which type of workload. The fraction of the total CPU service units consumed by each type of work was calculated and represents the percent of utilized CPU time consumed by each. Likewise, the fraction of total I/O service units represents the fraction of all I/O on the system imposed by each workload type.

Figure 2 shows the fraction of CPU utilization for each workload type. All customers had some batch work during the prime shift. Customers 11, 41, and 51 had a significant amount of TSO load. Complexes 61, 62, and 63 were predominantly CICS on-line production, and the rest were primarily IMS.

Figure 3 shows the fraction of EXCPs issued by each type of workload. The batch component of the workload tends to be more I/O intensive than the on-line applications. This is seen in the larger fraction of I/O consumed by batch than the CPU component.

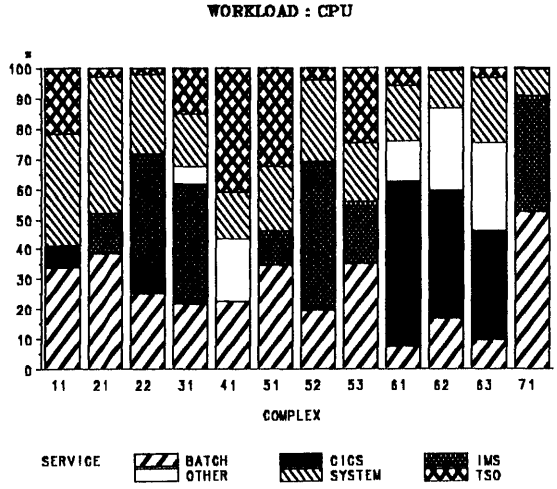


Figure 2. Percent of CPU Utilization by Workload Type

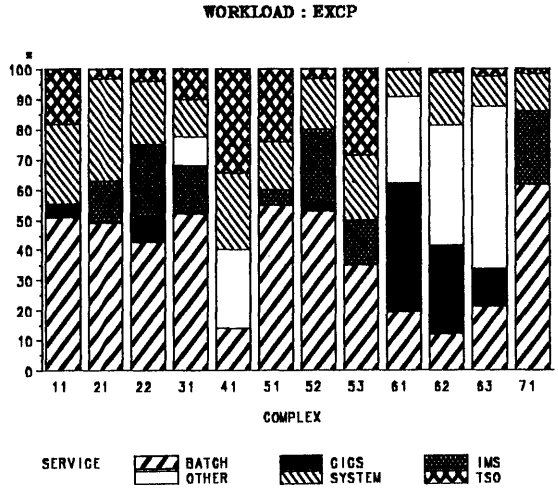


Figure 3. Percent of EXCPs by Workload Type

Consistently Very Active Data Sets

Method of Identification

The morning and afternoon GTF SIO trace data were processed with the VTOC dumps to associate each I/O with a particular data set. The data was processed to produce a list of all data sets that were active during each of the two 15-minute peak intervals. The data sets in the list were ranked by I/O rate. Each customer was asked to describe the use of their ten percent most active data sets. Additionally each customer identified a subset of these top ten percent that were always active during the prime shift, and whether any of these data sets were read-only as opposed to receiving update or write activity. Those data sets in the ten percent most active that the customer noted as being consistently active during prime shift became the consistently very active data sets for this study.

The results of the morning traces were compared to the afternoon traces to ensure that all data sets claimed to be consistently very active were indeed active during both trace periods. The fractions of total I/Os going to the consistently very active data sets in the morning and afternoon were compared and found to be essentially equal. For simplicity in processing, the morning traces were used for subsequent analysis.

The consistently very active data sets were identified and classified into the following four categories:

1. Key System: system control data sets, such as RACF access control, HSM control data sets, JES2 checkpoint, TSO UADS and BROADCAST data sets, Catalogs, SMF data, logs, etc.

2. Spool data sets: both the JES2 and JES3 spool files.

3. Libraries: including IMS program libraries, IMS format and ACB libraries, subroutine libraries, special product libraries, user application load libraries, TSO command libraries, procedure libraries (PROCLIB), and so on.

4. User data: application data sets, such as on-line data bases and indices or application files.

Once identified, there were several kinds of information that could be extracted from the data available for these data sets. The information was available at the data set level and could also be summarized for each of the four categories.

Results by Category

The first item of interest was what fraction of the total non-paging I/O, and what fraction of the non-paging space were due to each of the four categories of consistently very active data sets. Any reference in this paper to percent of I/O is a reference to the percent of non-paging I/O. The percent of I/O and percent of space allocated for each of the four categories of data is shown in Figure 4. The set of bars on the left shows the breakdown for I/O activity and the set of bars on the right shows the breakdown for space. Each bar within the set represents one customer. The height of the bar shows the fraction of total non-paging I/O (or space) attributed to the consistently very active data sets. Within each bar is a further breakdown by data set category. The three installations with the least amount of consistently very active user data were TSO oriented. The fraction of total non-paging I/O that was imposed by the consistently very active data sets ranged from around 25 percent to almost 70 percent! On the other hand the space occupied by these data sets was only 3 to 20 percent of the total non-paging DASD capacity. Of the four categories, the program libraries and key system data sets generally used the least space and had the highest I/O rates.

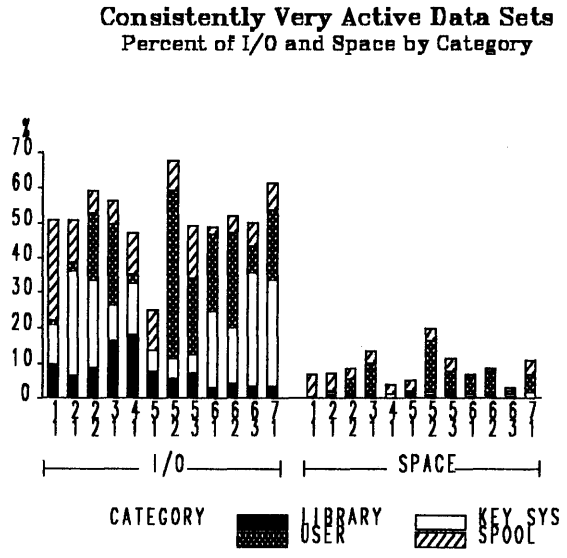


Figure 4. Percent of I/O and Space by Category for All Complexes

The I/O rate and megabytes allocated are shown in Figure 5 along with the percent of non-paging space and percent of non-paging I/O for each category of consistently very active data for each customer. The actual number of megabytes allocated to all the consistently very active data sets varied among customers. The range was from 4,463 MB to 18,649 MB. The total I/O rate to all of these data sets ranged from 104 to 468 I/Os per second.

Complex	Category	MB Allocation	I/O Rate	% of NP Space	% of NP I/O
11	Libraries	116.2	36.0	0.17	9.6
	Spool	4386.8	107.3	6.39	28.6
	Key System	244.6	43.1	0.36	11.5
	User Data	6.3	3.7	0.01	1.0
	TOTAL	4753.8	190.1	6.93	50.7
21	Libraries	681.5	25.7	0.87	6.5
	Spool	3769.4	48.5	4.80	12.3
	Key System	626.8	115.4	0.76	29.3
	User Data	580.4	10.1	0.74	2.6
	TOTAL	5658.0	199.7	7.17	50.8
22	Libraries	967.8	54.2	0.80	9.1
	Spool	3768.6	38.5	3.13	6.4
	Key System	553.3	146.2	0.46	24.5
	User Data	4945.1	114.0	4.10	19.1
	TOTAL	10234.9	352.8	8.49	59.0
31	Libraries	514.2	60.8	0.93	16.4
	Spool	1884.3	25.4	3.41	6.8
	Key System	178.4	37.3	0.32	10.1
	User Data	4877.9	85.1	8.83	22.9
	TOTAL	7454.8	208.6	13.50	56.2
41	Libraries	390.8	99.2	0.33	18.1
	Spool	2852.3	62.2	2.42	11.3
	Key System	1042.0	79.9	0.89	14.6
	User Data	178.0	15.3	0.15	2.8
	TOTAL	4463.2	256.7	3.79	46.7
51	Libraries	2182.5	59.4	1.95	7.6
	Spool	3133.4	88.2	2.79	11.3
	Key System	226.8	48.4	0.20	6.2
	TOTAL	5542.7	196.0	4.94	25.0
52	Libraries	1098.8	33.8	1.15	5.3
	Spool	3133.4	53.1	3.29	8.4
	Key System	427.3	39.0	0.45	6.2
	User Data	14034.6	300.0	15.00	47.4
	TOTAL	18694.1	426.0	19.90	67.3
53	Libraries	804.3	15.2	1.59	7.2
	Spool	1882.2	32.3	3.72	15.3
	Key System	64.8	11.1	0.13	5.3
	User Data	3078.6	45.1	6.08	21.4
	TOTAL	5829.9	103.7	11.51	49.2
61	Libraries	872.2	31.2	0.40	3.2
	Spool	1882.2	19.9	0.87	2.1
	Key System	1891.7	207.7	0.88	21.6
	User Data	10406.5	208.8	4.82	21.7
	TOTAL	15052.6	467.6	6.97	48.5
62	Libraries	829.0	29.7	0.41	4.2
	Spool	1882.2	36.4	0.93	5.1
	Key System	952.0	115.0	0.47	16.2
	User Data	13585.1	188.1	6.73	26.4
	TOTAL	17248.3	369.2	8.54	51.8
63	Libraries	891.2	15.8	0.47	3.3
	Spool	1882.2	32.3	1.00	6.7
	Key System	1363.3	155.3	0.72	32.3
	User Data	1647.5	36.1	0.87	7.5
	TOTAL	5784.1	239.6	3.07	49.9
71	Libraries	314.5	13.9	0.19	3.6
	Spool	6917.0	29.7	4.17	7.6
	Key System	2649.6	117.9	1.60	30.2
	User Data	8266.3	77.9	4.98	19.9
	TOTAL	18147.4	239.5	10.94	61.3

Figure 5. Summary of Consistently Very Active Data by Complex

Static vs Dynamic Management

Method of Analysis

The consistently very active data sets were identified in the previous section and their size, I/O rates, and usage were described. In this section the focus will be on how dynamic management of these data sets compares with static management. For simplicity the analysis assumes that the electronic store is always non-volatile. In other words, no distinction needs to be made between reads and writes. This mythical non-volatile electronic store that can be managed either statically or dynamically is created for the purpose of the analysis presented in this section. It allows a focus on the locality of

reference of consistently very active data sets. Dynamic management and static management were compared for all data sets by category.[1]

Static management of the consistently very active data was evaluated by assuming that these data sets were being placed in a static store in the best possible order. The highest access density data sets would go in first. The access density is the I/Os per second per megabyte of space allocated to the data set. The amount of I/O handled out of the static store was simply the amount of I/O that went to the data sets residing in a given-sized store.

The ranking of the data sets by access density was based on the morning SIO trace. The I/O handled by the static store was calculated by adding the I/O rates for all of the data sets that fit in a given memory size. These I/O rates are also based on the morning SIO trace. Although the consistently very active data sets have relatively stable I/O rates, there is some variation from one time of day to another. For this analysis the data sets were chosen for the store in an optimal order for the morning I/O trace; variation in the I/O rates would change the amount of I/O handled for any given memory size from what is shown here. [2]

Dynamic management of the consistently very active data sets was evaluated by assuming that they were the only data under a single dynamically managed electronic store with a varying amount of memory. The hit ratios were determined as a function of the size using the Cache Analysis Aid (CAA) simulation program. A special version of the program was used that accepts GTF SIO trace data rather than the more complete CCW traces.[3] This version treats all I/Os the same (rather than distinguishing between reads and writes) and predicts an overall hit ratio. This overall hit ratio is the hit ratio one would expect from a non-volatile store. [4] The management algorithm simulated was similar to the 3880 Model 23 in that it would stage the record accessed and the remainder of the track into the memory, and remove tracks from memory on a least recently used basis. Input to the simulation was created by first merging together the GTF SIO trace data from multiple systems sharing DASD (i.e., from a whole complex), and then selecting only those SIO records for activity going to the consistently very active data sets. The resulting trace records were further divided into four trace files, each containing only I/O for one of the categories. The CAA program was then run to determine the hit ratios as a function of memory size for each of the four categories as well as all of the consistently very active data sets together.

The percent of non-paging I/Os captured in memory was used to compare the performance of static versus dynamic management. The comparisons were made based on equal amounts of memory. For static store the total percent of I/Os captured is the accumulated percentage of I/Os captured by all data sets that fit in a given memory size. For dynamic management, the percent of I/Os captured is the hit ratio multiplied by the total I/O rate of all the consistently very active data sets.

Results

A set of five graphs was produced for each of the twelve complexes studied. Included were 1) the hit ratio for the four categories of data as a function of the amount of memory, and 2) the percent of total non-paging I/O handled out of varying sized stores when managed statically and when managed dynamically; this graph was generated for libraries, key system data, user data, and spool.

Unfortunately, 60 graphs is a bit overwhelming. Since the pattern in the results was strikingly similar in all customer installations, only one example is shown here. Complex 53 was chosen for display. Figure 4 shows complex 53 as having more user and spool data accesses than key system and library data. This is not unusual for production on-line systems, in this case IMS.

Figure 6 shows the hit ratios of each of the four categories of consistently very active data and the hit ratios of all this data managed together for complex 53. The overall hit ratio is shown on the vertical axis, and the MB of cache memory is shown on the horizontal axis.

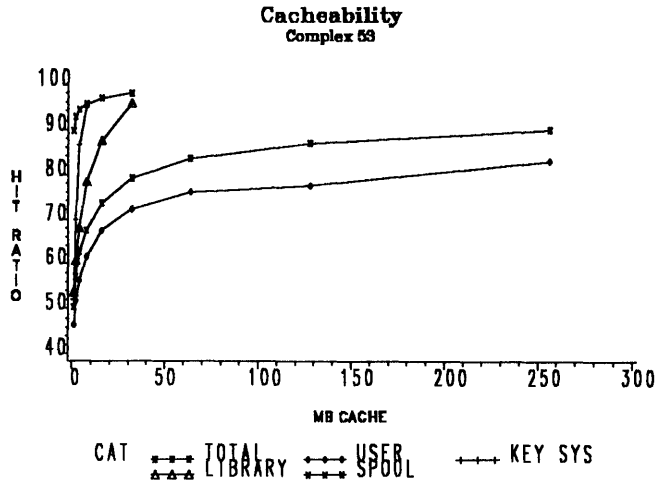


Figure 6. Hit Ratios by Category for Complex 53

The best locality of reference, as indicated by very high hit ratios for small amounts of cache memory, is seen in the spool data. Key system control data and libraries also have very good locality of reference. User data tends to have a more random reference pattern resulting in lower overall hit ratios for a given amount of cache than the other categories. The overall hit ratios for complex 53 are heavily influenced by the locality of reference in the user data; recall that this IMS production system had fewer key system and library accesses than a TSO installation.

The only category of data that could be considered read only is the Libraries. The other categories all have update activity. Spool data tends to be almost half write activity. Key system control data is predominantly read. User data is usually predominantly read, but this varies from customer to customer. If the electronic store is non-volatile, the fraction of write activity is not a concern. The thing to note here is that only the libraries are suitable for a purely volatile store.

Figure 7 through Figure 10 show the comparison of static vs dynamic management for each of the four categories of consistently very active data at complex 53. The percent of total non-paging I/O that is captured in the electronic store is shown on the vertical axis. This axis is scaled the same in all four graphs. The MB of storage is shown on the horizontal axis. This axis has the same scale for key system and library data, but goes to much higher amounts of memory for the user and spool categories.

---

1 In the section on the application of the analysis to non-volatile cache compared to a solid state device (SSD) realistic assumptions are made for these devices and some of the consistently very active data sets are removed from consideration.

2 In the more realistic analysis of cache vs SSD the amount of variation seen in the afternoon sample will be discussed.

3 The authors would like to acknowledge the extensive help provided by J.S. Hyde, the developer of the IBM Cache Analysis Aid, in making the analysis in this paper possible.

4 In the analysis of cache vs SSD assumptions are needed and made about read to write ratios.

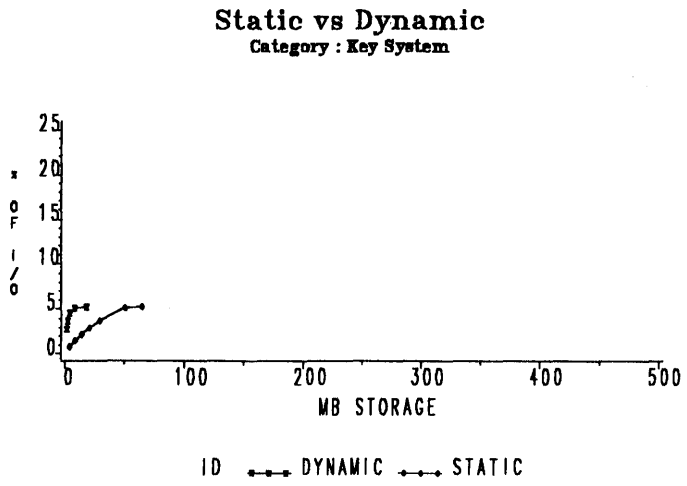


Figure 7. Static vs Dynamic Management for Key System Data

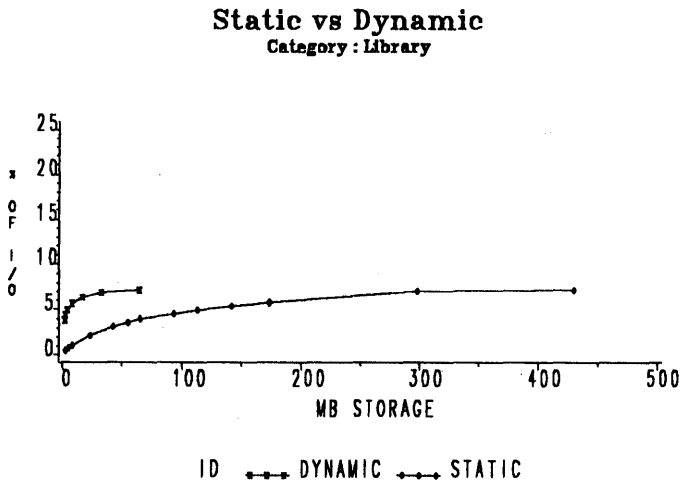


Figure 8. Static vs Dynamic Management for Library Data

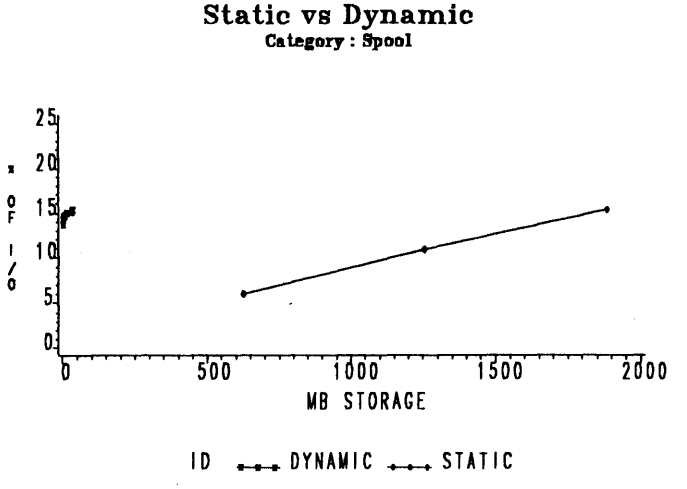


Figure 9. Static vs Dynamic Management for Spool Data

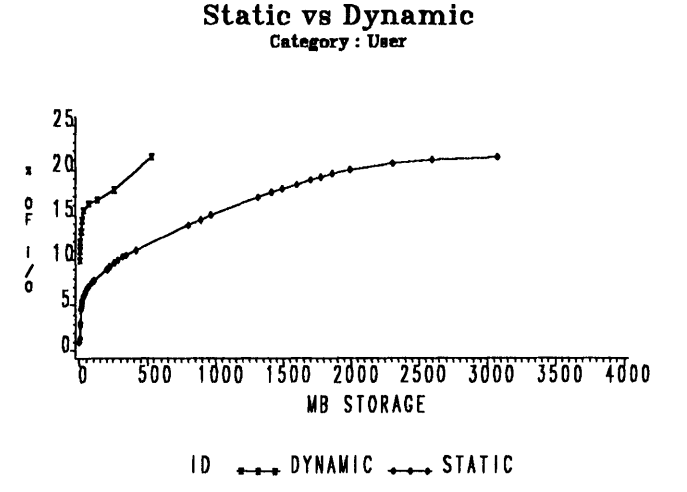


Figure 10. Static vs Dynamic Management for User Data

The dynamic management of the consistently very active data sets was better than static management in all four categories. Much less memory was required to handle the same amount of I/O at memory speed when the memory was dynamically managed. One reason for this is that, even over a time period as long as 15 minutes, the total number of bytes referenced in a data set is much smaller than the total number of bytes allocated to the data set. Another way of looking at these results is to note that for the same amount of memory, the dynamically managed data provided a much higher fraction of the total non-paging I/O at memory speed than did the statically managed store.

Application of Analysis to Cache vs SSD

Motivation

The above analysis provides some insight into the locality of reference of different categories of very active data, but does not make it easy for an installation to evaluate the relative effectiveness of a cached control unit and solid state device (SSD). It is not practical to place only consistently very active data sets under a cache. In practice an installation is more inclined toward placing whole volumes of data on a cache control unit. Furthermore, SSD is expensive; it is not realistic to put all the consistently very active data sets in SSD.

Spool data sets in particular have a low return on investment. They are too large for their I/O rate to be cost effective in an SSD, and have too low a read to write ratio to cache effectively on cache control units with volatile memories in spite of their good locality of reference. It is not even clear that speeding up the spool data set I/O would improve customer perceived performance. Thus, it seems that a customer would not seriously consider spool data sets for any kind of electronic store. Similarly, IMS logs are not good candidates for cached devices because of the predominance of write activity. They are not good candidates for SSD because of their size.

VTOCs are a special case of consistently very active data. They must remain on the volume with the data sets they point to. It does not make sense theoretically to place several VTOCs on an SSD. Thus, VTOCs were not considered as candidates for SSD.

Even with spool, IMS logs, and VTOC data eliminated from consideration, some customers have more consistently very active data than can realistically be placed on an SSD. Usually, the least I/O intensive of this data is some portion of the user data category.

A modified analysis was undertaken that was designed to reflect customer requirements realistically. Because the customers varied considerably in the fraction of non-paging I/O that goes to very active data, a form of normalization was introduced to facilitate the generalization of conclusions.

Analysis Method

The process of determining the SSD candidate data sets began with including the consistently very active data sets of all categories except spool. Then, in addition to spool data sets, the IMS logs and VTOCs were eliminated for the reasons outlined above. The SSD was assumed to be a non-volatile electronic store, so data sets with update activity were included as SSD candidates. Some installations had replicated IMS program libraries. These were aggregated by accounting for all of the I/O, but accounting for the space only once. In other words, only one volume was cached and only one copy of the data set was placed in the SSD, but all of the I/O was assumed to go to the one copy.

The criterion used to place data sets into the Solid-State Disk was based on access density (IOs/SEC/MB). The data sets with higher access densities were put into SSD first, until accumulated allocated space was over 2000 MB (a reasonable limit for SSD). The data sets that fit within the 2000 MB limit formed the set of SSD candidates.

In the case of cache, entire volumes were placed under the cache as opposed to only the SSD candidate data sets. Any volume that contained one or more of the data sets that would go into the 2000 MB SSD was placed on a cache control unit. The VTOCs associated with these cached volumes were of course cached. The cache was assumed to have volatile memory, and the cache management algorithms were assumed to be those of the 3880 Model 23.

Performance of the cache control unit was compared to the performance of the solid state device based on the amount of I/O that was handled out of the memory for components with equal memory and also for equal cost. The I/O handled out of the SSD was the total I/O to those data sets that would fit in the particular amount of SSD memory.

The amount of I/O handled out of the cache memory was the total I/O rate to all of the cached volumes, times the hit ratio, times the fraction of the I/O that were reads. Since the GTF traces used for cache analysis contained no CCW information, the read hit ratios are not exact and some assumptions need to be made about read to write ratios. In a separate calibration study several GTF CCW traces from some customers not included in this study were used to obtain overall hit ratios and read hit ratios. For all practical purposes the two hit ratio measures were the same. The overall hit ratios determined in this study from the SIO traces were used with confidence to predict the read hit ratios. Read to Write (R/W) ratios needed to be estimated from data collected for other studies. The studies described in the Washington Systems Center Technical Bulletin, IBM 3880 Model 23 Performance Measurements (order number GG66-0223) published in January of 1986, were used to estimate the read to write ratios for the categories of data evaluated here. The program library data sets were read only. The key system data sets had very high read to write ratios and user data tended to have R/W ratios of around 6:1. A ballpark estimate of an overall R/W ratio of 9:1 was chosen for the data considered in this analysis.

Results

The first set of results shows how much of the total non-paging activity on each system goes to 1) the SSD candidate data sets and 2) the volumes on which SSD candidate data sets reside. This information is needed in order to keep a perspective on what fraction of all I/O is being evaluated for management in some type of electronic store, and how this fraction varies depending on the type of installation. The leftmost set of bars in Figure 11 shows the percent of total non-paging I/Os that went to the SSD candidate data sets (solid part of bars) and to the cached volumes containing those data sets (total bar height). Each bar represents a different customer complex. With the SSD candidate selection used, about 17 to 45 percent of the non-paging I/O went to data that would fit into a full 2000 MB SSD. The percent of non-paging I/O that would go to the cached volumes ranged from about 20 to 60. The variation was due to the type of workload. The customers on the low end of these ranges were TSO environments where there were virtually no consistently very active user data sets. The customers with the higher fraction of I/O going to the SSD candidate data sets generally had production on-line application data included.

The rightmost set of bars shows the percent of the total non-paging DASD space allocated to the SSD candidate data sets (solid part of bar), and to the volumes under the cache (total bar). The variation here is due for the most part to the total number of volumes that needed to be included in order to account for all of the SSD candidate data sets.

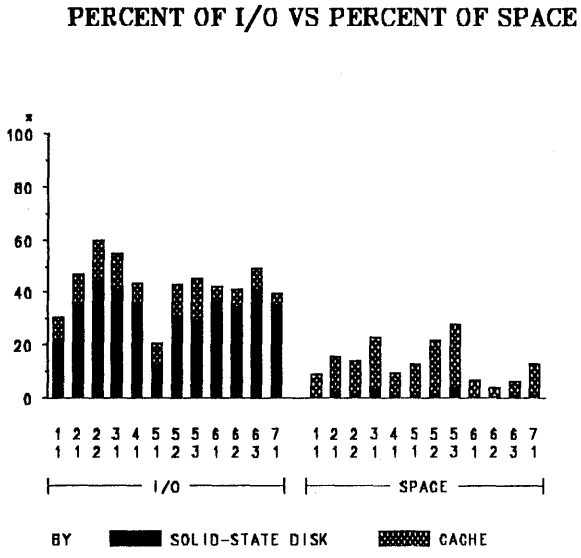


Figure 11. Percent of I/O and Space

The amount of I/O handled by a solid state device of increasing size is shown in Figure 12. Each customer installation is shown with a separate line on the graph. The graph has been normalized so that the vertical axis value of one represents all of the I/O going to the SSD candidate data sets. Recall from Figure 11 that this amount of I/O is a different fraction of total non-paging I/O for each installation. For all customers the curve rises steeply at first and then levels off as data is put into the SSD that has less return on investment.

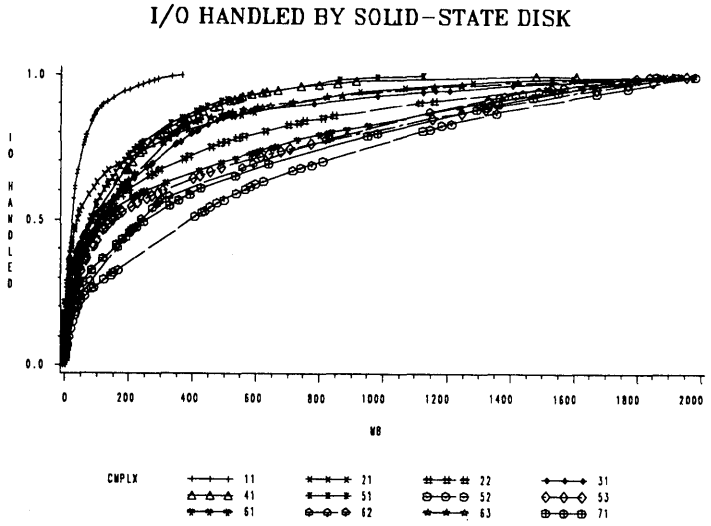


Figure 12. I/O handled by the Solid-State Disk for Each Complex

The issue of variability in the I/O rates was raised in the earlier discussion of static vs dynamic management of all of the consistently very active data sets. An analysis was done on the variability in the I/O rates to the SSD candidate data sets. The afternoon trace was analyzed to determine what I/O rates occurred to the data sets that were chosen to go into the SSD based on the morning traces. The percent of all non-paging I/O going to the SSD candidate data sets in the afternoon was calculated as a function of SSD size. The data sets were still assumed to be packed into the SSD in the order derived from the *morning* traces.

Figure 13 shows the results for the three TSO intensive complexes, and Figure 14 shows the results for three of the IMS customers. A subset of the customers is shown for example. The vertical axis shows a simple percent of non-paging I/O going to the data sets that fit within the SSD memory size shown on the horizontal axis. The solid line shows the morning percent, and the broken line shows the afternoon result. The three TSO customers vary little from the morning to the afternoon. Two of the customers obtained a higher percent of their I/O out of the SSD during the afternoon sample and one obtained a lower percent. The average change in percent of non-paging I/O for the TSO customers was .3 percent more I/O in the afternoon sample. The IMS customers varied more than the TSO customers. Note that these customers have more user application data in their SSD than the TSO customers do. The three customers shown in Figure 14 are the best IMS case, the worst IMS case, and one in between. The overall average change in percent of non-paging I/O for the IMS customers was 7.1 percent less I/O captured in the afternoon.

This sample is too small to draw sweeping conclusions. There appears to be no significant variation in the fraction of I/O handled by the SSD in TSO environments. In IMS installations, the variation averaged 7.1 percent and in one case was as much as 15 percent less I/O for the afternoon sample. It appears that key system and library data sets have consistent enough I/O rates from one time of day to another that an SSD optimized from one trace is likely to remain reasonably well optimized. On the other hand, on-line application data that is consistently very active still appears to have a lot of variation in I/O rates from one time of day to another. An SSD that is optimized for one time may yield less I/O during another time. In this analysis we are optimizing the placement of data sets in the SSD using the morning sample. Since we also use the I/O rates from the morning sample to compare the performance of the cache and the SSD, we are giving the SSD the benefit of the doubt about I/O rate variability. In customer sites where on-line application data resides in the SSD, the SSD may perform less well than is shown in this analysis.

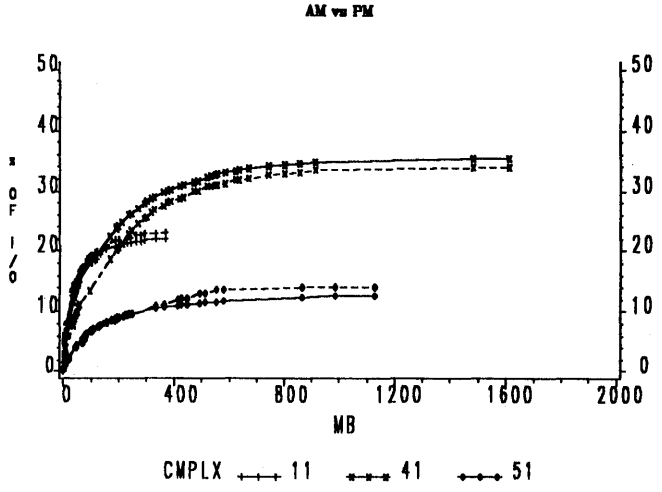


Figure 13. AM vs PM Activity for TSO Complexes

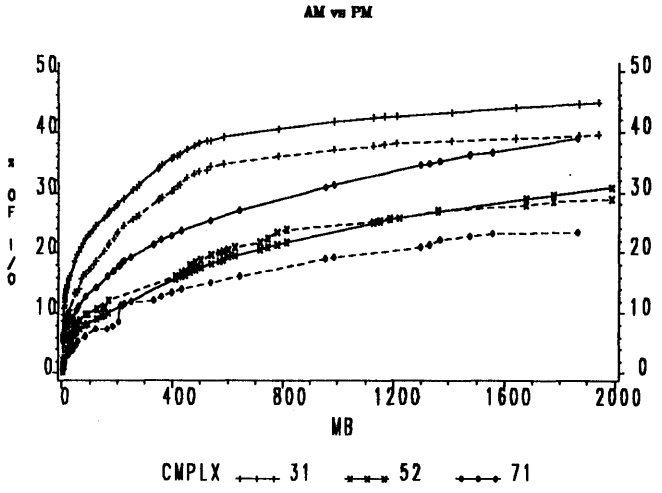


Figure 14. AM vs PM Activity for IMS Complexes

Figure 15 shows the amount of I/O handled out of increasing amounts of cache memory. For memory sizes over 64 MB, assume that more than one 3880-3 control unit was upgraded to a 3380-23. In most customer installations about 64 MB of cache provided most of the available benefit. The vertical axis is normalized in the same way as in the Figure 12: the value of one represents the amount of I/O that could be obtained out of a 2000 MB SSD. The fact that there is more data under the cached control unit than there is on the SSD allows the amount of I/O handled by the cache to go over the value of one. The fact that not all I/Os are read hits tends to drive the number down. All but one customer was able to handle at least as much I/O from a moderate amount of cache memory as was handled by a 2000 MB SSD. The range was from about 80 percent to 150 percent of the SSD I/O.

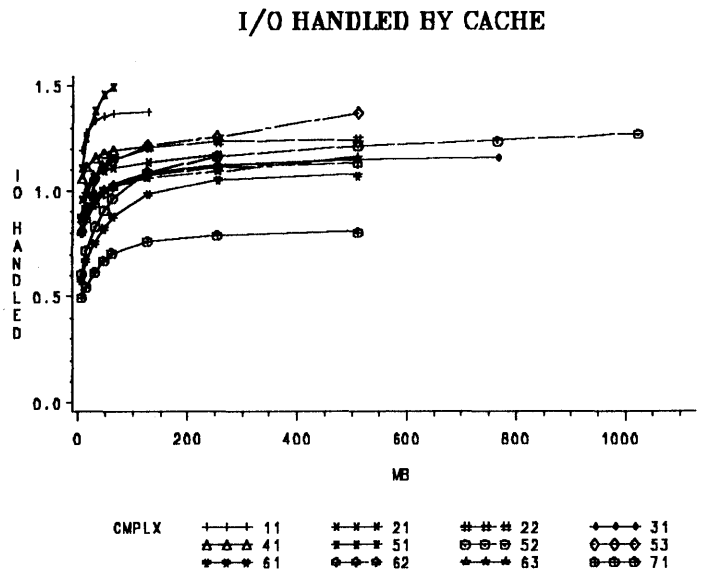


Figure 15. I/O handled by the Cache for Each Complex

If each installation's SSD curve from Figure 12 were paired with the respective cache curve from Figure 15, the SSD curve would fall under the cache curve. In other words, for the same amount of electronic memory, the cache is able to handle more I/Os out of the cache memory than are handled out of the SSD. Figure 16 shows the average cache curve and average installation SSD curve. At 64 MB of memory, the cache control unit handles about 2.5 times more I/O at electronic speed than the SSD does. At about 200 MB of memory, the cache handles about 1.75 times more I/O at electronic speed than the SSD does.

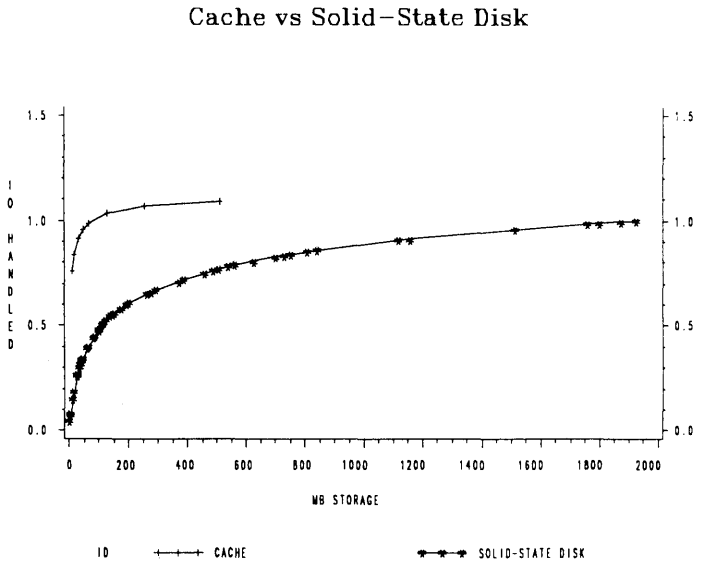


Figure 16. I/O handled by Cache and SSD for Equal MB Comparison

The more interesting question, of course, is how do the cache and SSD compare on an equal cost basis. The starting point for addressing this question is to make some cost assumptions. Prices change in time and vary with circumstances. Pricing assumptions used in this analysis are for illustration and are somewhat arbitrary; the numbers should be adjusted to reflect other situations. It was assumed that the cost of the cache would be the cost for the MES upgrade of a 3880-3 control unit to a 3880-23. This makes sense since there would be no need for additional control units, only the adding of cache to one or more existing control units. The price used for the MES upgrade was $321,130 dollars for the 64 MB unit. The 8 MB upgrade was priced at $69,130 dollars. Upgrades between 16 and 64 MB were base priced at $33,130 plus $72,000 for each 16 MB increment. Price assumptions for the SSD were more arbitrary since there is a range of prices depending on vendor and discount. The prices used reflect an example of prices for non-volatile SSDs that are heavily discounted from their list prices. The example price includes both controller and memory. An entry level system was assumed to be 64 MB and to cost $195,000. Memory increments over the base system were priced at $1,000 per MB. The maximum single unit was assumed to be 512 MB for a cost of $643,000.

If these cost assumptions are applied to the MB of storage given in Figure 16, then the I/O handled can be shown as a function of the cost. Figure 17 shows the I/O handled out of cache memory and the I/O handled out of SSD with cost on the horizontal axis. The entry level cost for the cache is less than the cost for an entry level SSD, and the cache handles almost twice the I/O handled by SSD system. If the chart is examined to determine a point that might be considered optimal for the cache and a point that might be considered optimal for the SSD, a cost of about $300,000 comes to mind. Beyond this point both types of devices offer little return on investment. A $300,000 investment would yield about 1.7 times more I/O out of the cache than out of the SSD.

In summary, when reasonable assumptions are made about data to put into an SSD and how to get the same data under a cache control unit, the cache solution looks better. The cache handles about twice as much I/O out of a reasonably sized cache as the SSD does with a similar amount of memory. The same I/O rate out of cache can be obtained with a *much* smaller memory size than would be needed for an equivalent I/O rate out of the SSD device. A comparison of the cache and SSD performance based on equal cost configurations also shows the cache to be the better solution. At configurations of about $300,000, the cache handles about 1.7 times as much I/O out of memory than the SSD does.

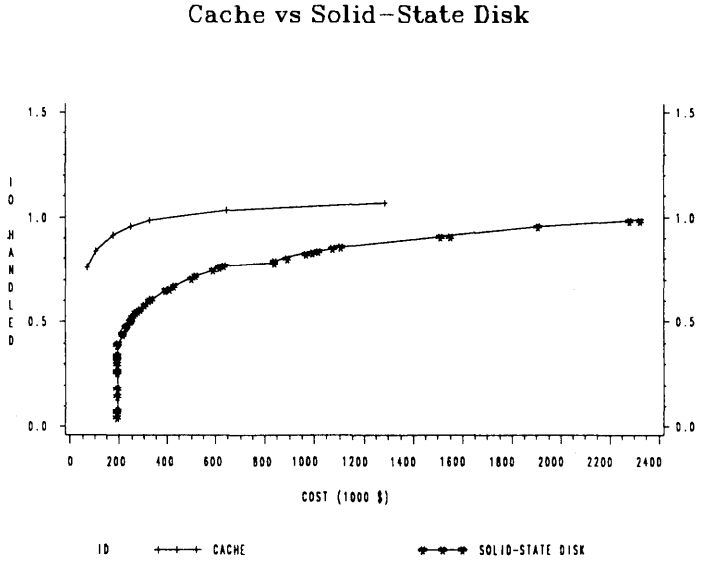


Figure 17. I/O handled by Cache and SSD for Equal Cost Comparison

Conclusions

- The consistently very active data sets in the customer complexes studied were responsible for about 25 percent to 70 percent of the non-paging I/O on the system. There was less I/O to this type of data in TSO installations than was seen in IMS or CICS dominated installations because of the absence of on-line application data that is active every day.
- Libraries and key system control data sets had the highest I/O rates for the amount of storage they occupied. Spool data and some user application data were consistently very active, but less I/O intensive.

- Libraries, key system data and spool data had a higher locality of reference than user application data.
- Libraries were the only category of consistently very active data that were read only. Spool data in particular has a low read to write ratio.
- All four categories of data had good enough locality of reference that a dramatic advantage was seen in managing this data dynamically rather than statically.
- When the subset of the consistently very active data sets that are suitable for placement in a Solid State Device (SSD) was selected, they accounted for about 17 to 45 percent of the non-paging I/O on the system.
- The volumes containing the SSD candidate data sets had I/O rates that accounted for about 20 to 60 percent of the non-paging I/O on the system.
- For reasonably sized cache and reasonably sized SSD, the cache control unit handled about twice as much I/O out of cache memory as was handled out of the SSD memory.
- For an equal cost cache and SSD, the cache control unit handled about 1.7 times more I/O than was handled by the SSD.

A Simulation Model for Determining Optimal Freespace Levels on MVS Disk Storage Volumes

Kenneth E. Levy
Mobil Administrative Services Company Inc.
P. O. Box 1033
Princeton, New Jersey 08540

ABSTRACT

Freespace within MVS disk storage pools is a necessary evil. Without the proper amount of freespace, users will be subjected to application failures due to lack of disk space. This paper describes a simulation model for determining the cost/benefit tradeoff for maintaining various levels of freespace on MVS disk storage volumes. Economically sound policies for freespace levels can then be derived from the results.

INTRODUCTION

Elimination of private disk ownership and establishment of system managed pools of disk space is justified for the labor savings derived from having the computer automatically control and maintain storage[1]. Pooled storage does pose new problems and costs, however. Instead of humans allocating and scratching datasets and manually controlling freespace, the system must perform this function. For new datasets, the system performs this function quite well. The system will automatically search all volumes in a storage pool to locate a volume with sufficient space to satisfy the request. Finding space for secondary extents for existing datasets, however, presents a problem. These requests must be satisfied from freespace on the volume on which the data set resides. Thus, freespace must be maintained on volumes to minimize the number of abends due to dataset extension failures. The storage administrator is faced with the question of the optimal level of freespace to be maintained on an MVS storage volume? Is it ten percent? Twenty percent? Or is it even larger? Clearly it is not cost effective to maintain disk space which is never used. It is also uneconomic to reduce freespace to a level which generates abends (due to cost of reruns for failed applicatiosn). This paper will present a method for solving the optimal freespace problem.

ANALYTIC MODEL SOLUTION

Initially, an analytic model was proposed to address this question. It was thought that if the extent size frequency distribution could be fit to a curve, a probability function could be derived to determine the probability of allocation failure for any given level of freespace. Unfortunately, when the size frequency distribution was examined (Figure 1), it presented several problems. First, it was impossible to fit a smooth curve to the distribution function. There were spikes in the curve at regular intervals (15 tracks, 30 tracks, 45 tracks, etc.). These spikes generally represent datasets allocated in units of cylinders rather than tracks. A second, third, or higher order equation could have been fitted to handle these discontinuities. However, this presented several new problems. Each pool, with its corresponding frequency distribution, would have to be fit with its own function. If a large number of pools were to be examined, the effort required would be considerable. Also, this method presents difficulties if there is an irregular pattern of spikes. For example, allocations could be bunched at one, five, and ten cylinders, but not at intervening cylinder boundaries. These are typical of allocation quantities "guesses", where the exact amount of storage was not pre-calculated. This form of model was finally rejected due to lack of a "general" solution which would fit all forms of frequency distributions (pools).

SIMULATION MODEL SOLUTION

A simulation model seemed to present a logical next step for tackling this problem. If the allocation / scratch process could be simulated by a program, the number of allocation failures could be counted and the probability of failures at various freespace levels could be calculated and plotted (Figure 2). The optimal freespace can be then be determined by simply inspecting the plots to determine the level of failures acceptable to a particular installation.

The simulation program which models the allocation / scratch process on the simulated volume is rather simple. Two lists are maintained, the first representing allocated extents on the volume, and the second representing freespace extents on the volume. Initially, the allocated list is empty, and the freelist contains one entry reflecting the total number of tracks contained on the simulated volume. As datasets are allocated, entries are added to the allocated list and the entries in the freespace list are manipulated, much like the FORMAT5 DSCB's on a MVS volume's VTOC.

To simulate reality, the sizes of allocated extents within an actual dasd pool were used to generate simulated data sets with the same size frequency distribution within the model. This was accomplished by converting the size frequency distribution (Figure 1) into a cumulative frequency distribution (Table 1). The number of tracks to be allocated for each simulated dataset is determined by using a random number to index into the cumulative percent column.

Cumulative Frequency Distribution

Tracks	Frequency Percent	Cumulative Frequency Percent
1	46.7008	46.7008
2	10.7002	57.4011
3	4.1500	61.5510
4	2.3394	63.8904
5	5.7170	69.6074
6	1.3139	70.9213
.	.	.
2655	0.0032	99.9968
3300	0.0032	100.0000

Table 1.

TSO POOL

FREQUENCY BAR CHART

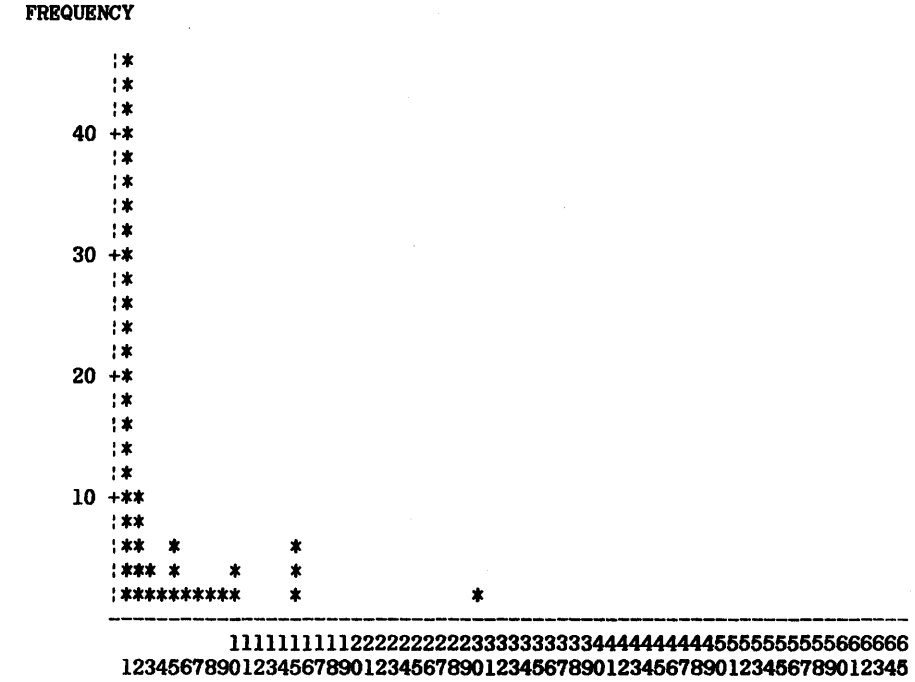


Figure 1.

TSO POOL

Plot of FAILPROB*PCALLOC Legend: A = 1 obs, B = 2 obs, etc.

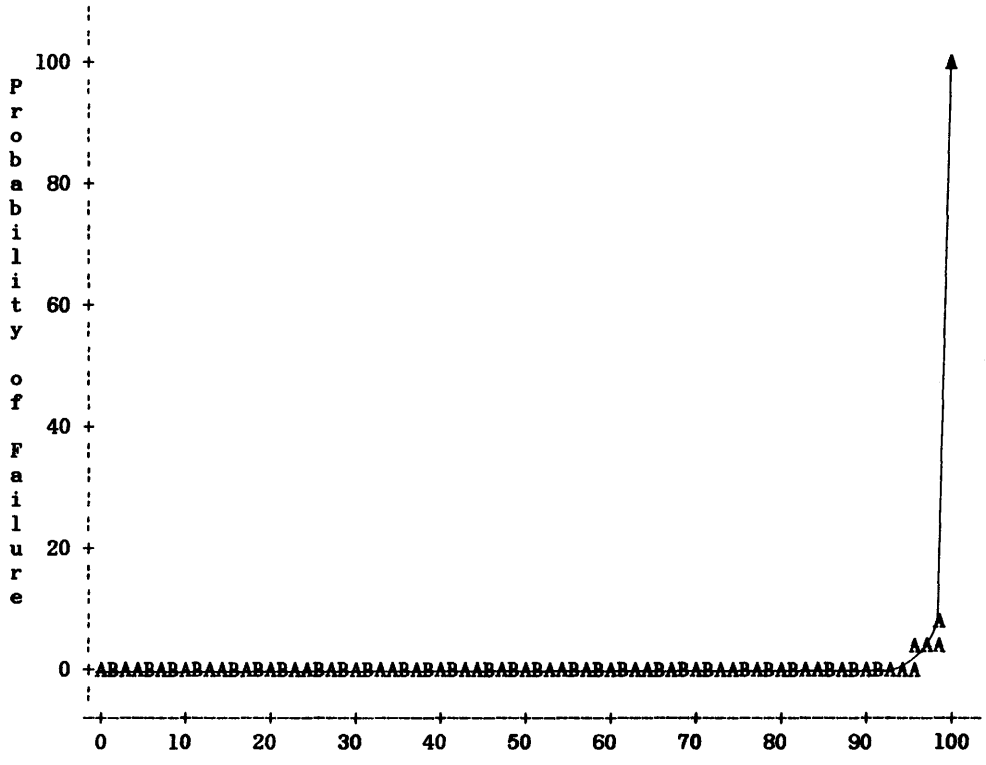


Figure 2.

To simulate the allocation of a dataset extent, a random number is generated between zero and one hundred percent. The cumulative distribution table is then scanned until the corresponding percentage is reached. The extent size that corresponds with that cumulative frequency will be used in the attempted allocation. As more and more datasets are allocated on the simulated volume, the frequency distribution of sizes on the simulated volume will closely correspond to that of the original pool. Parameterization of the simulation model for an existing pool of disks becomes simply a task of mapping the existing dataset extents on the volumes and converting the map into a frequency distribution. This makes parameterization of the model extremely simple for virtually any disk pools.

The simulation model was written in approximately 650 lines of MICROSOFT QUICK BASIC code, of which approximately one third are comments. The input to the model consisted of the cumulative frequency distribution curves, enhanced by the addition of cylinder allocation and unmovable attribute probability measures for each extent size. These last two items were used to more closely approximate the actual allocation process.

The allocation / scratch process was simulated completely in memory with no I/O required until the completion of the simulation run. The freespace on the simulated volume was varied from a target level of one percent to ninety nine percent (the endpoints of zero, where all allocations should fail, and one hundred percent, where no allocations should fail, are known). At each target level, datasets were allocated until the target freespace was reached. The success or failure of the allocations done during this "run-up" to the target level was not recorded. After the target was reached, one thousand allocations were attempted. After each attempt, the success or failure (due to space unavailable) is recorded. Datasets were then randomly deleted (if required) to back the volume down to the target freespace level. The number of allocation failures was accumulated in a matrix for conversion to a percentage and printing when all freespace levels had been simulated. The final output consists of the target freespace percentage and the probability of failure (number of failures divided by the number of attempts) at each level.

RESULTS

The model was run for three different frequency distributions reflecting three pools found at on large data center sharing disks between a complex of three 3090-200 processors. Several statistics were derived to help characterize the datasets within each pool (Table 2).

POOL	MEAN SIZE (tracks)	MEDIAN SIZE (tracks)	95%-tile (tracks)	STD DEV. (tracks)
TSO	11	2	45	56
BATCH	33	8	120	139
RESERVED	46	2	129	253

Table 2.

Each pool was simulated separately. Combined, the runs took approximately twenty-three hours of CPU time to run on a 6-Mhz IBM PC/AT with 640K memory. No extensive optimization or tuning of the model was performed. The frequency distributions and model results for each pool are presented in Figure 1 through Figure 6. They results are also summarized in table three.

	Percentage Freespace to be Maintained		
POOL	Probability of Failure < 1 %	Probability of Failure < .5 %	Probability of Failure 0 %
TSO	10 %	14 %	18 %
BATCH	22 %	27 %	48 %
RESERVED	25 %	28 %	56 %

Table 3.

It should be noted that for the pools with numerous large datasets, relatively large amounts of freespace are required to reduce to a zero percent probability of failure for allocations to that pool. This suggests that 'very large' datasets should be segregated from the small datasets. A relatively small amount of freespace would have to be maintained in a small dataset pool to minimize allocation failures.

CONCLUSION

A simulation model for determining optimal levels of freespace has been described. The model provides an approach that is easily adaptable to a large variety of dasd pools.

BIBLIOGRAPHY

[1] IBM, MVS/Extended Architecture Storage Management Library: Managing Storage Pools, (GC26-4264), 1986.

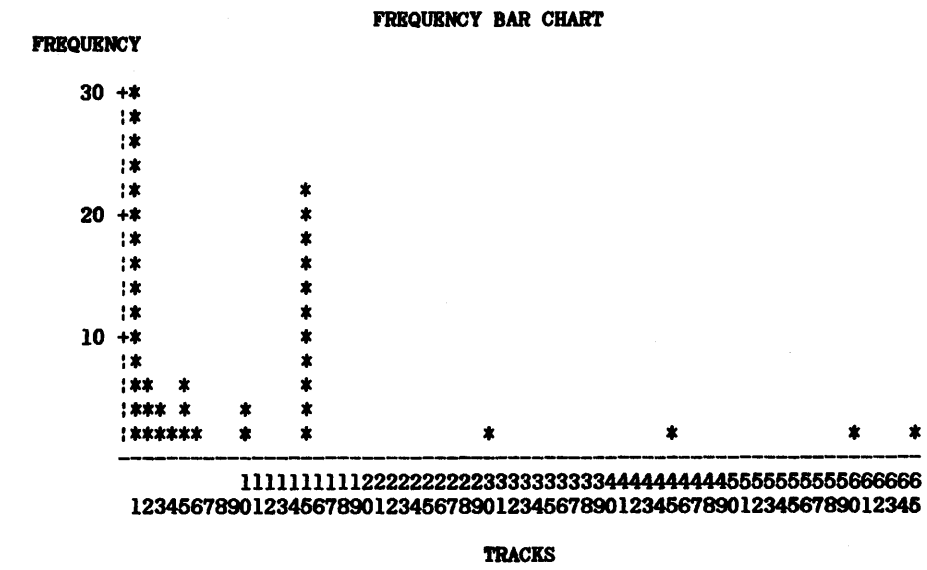


Figure 3.

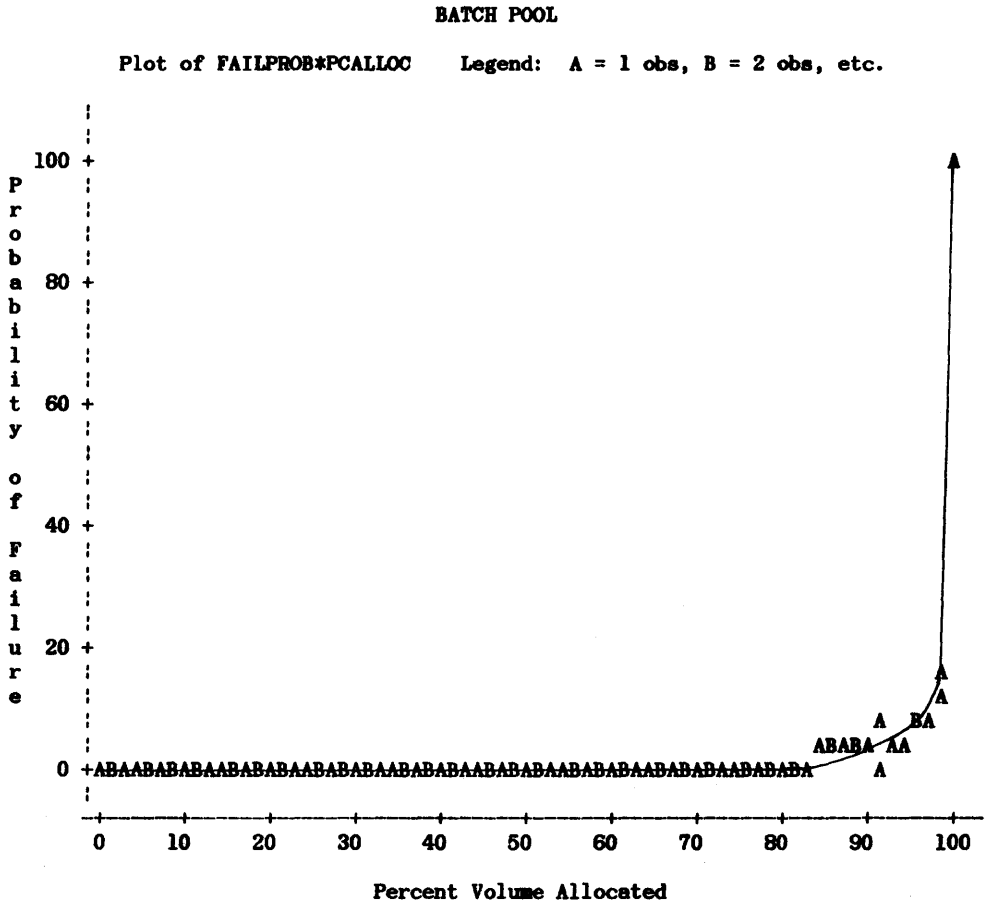


Figure 4.

RESERVED POOL

FREQUENCY BAR CHART

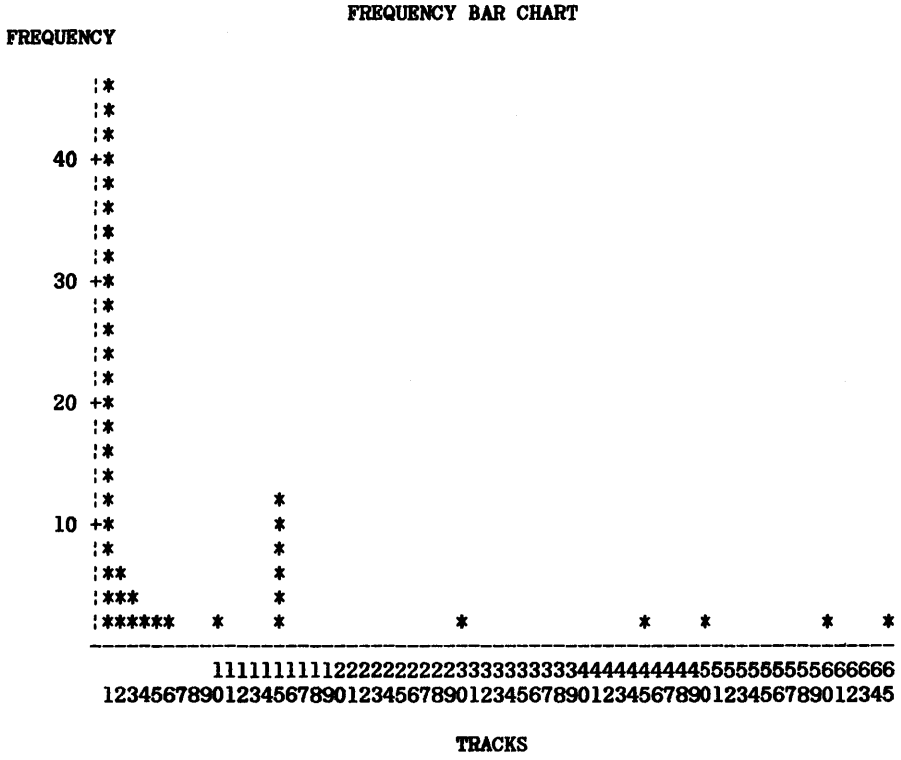


Figure 5.

RESERVED POOL

Plot of FAILPROB*PCALLOC Legend: A = 1 obs, B = 2 obs, etc.

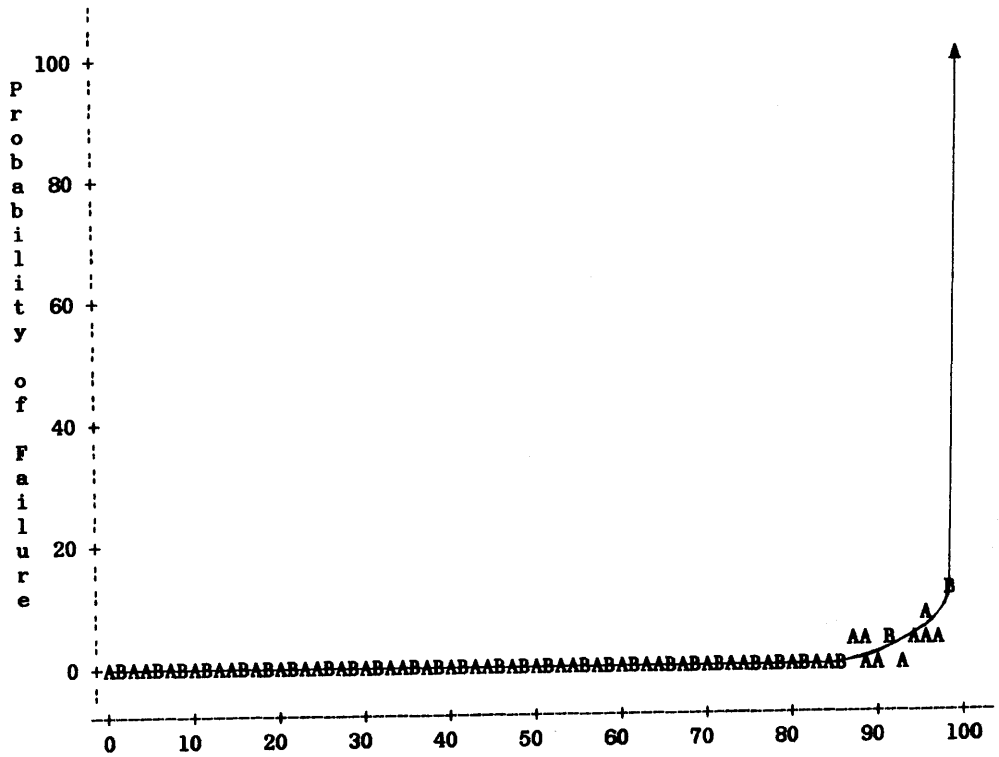


Figure 6.

NEW MEASUREMENT DATA SOURCES FOR MVS X/A AND ITS SUBSYSTEMS

H. W. Barry Merrill, PhD
Merrill Consultants

ABSTRACT

Since MVS/XA 2.1.0 became available in 1984, IBM has made significant changes in MVS/XA software and hardware architecture. New data exist to measure and manage these enhancements. SMF/RMF measurement of the vector processor, extended storage and the 3090 I/O processing enhancements will be discussed. Additional enhancements and the incompatible changes that have been announced in MVS/XA 2.2.1 will be described.

OVERVIEW

1. TYPE 70 CPU DATA
2. VECTOR FACTILITY MEASUREMENT
3. TYPE 71 EXTENDED STORAGE ARCHITECTURE
4. TYPE 72 WORKLOAD MEASUREMENT
5. TYPE 74 DEVICE MEASUREMENT
6. TYPE 78 I/O FOR 3090
7. TYPE 78 VIRTUAL STORAGE
8. MVS 2.2 ANNOUNCEMENT
9. 1 MIPS 2 MEG

1. TYPE70

RMF CPU Activity

There have been several significant changes to TYPE70 since 1984.

What is a CPU? Originally, a CPU was the Central Processing Unit; for some time, we have had multiple CPUs executing under the control of one operating system, and they have been called MPs, Dual Processors, APs, Dyadics, Triadics, Quads, IPL and non-IPL, and so forth. In some papers, this author has used the term multi-engine system to describe these complexes in which more than one CPU is under the control of a single copy of the operating system. Recently, IBM has introduced another term, the Central Electronic Complex (CEC), to describe the physical hardware that contains one or more CPUs (processors, engines) under the control of a single copy of the operating system. Take your pick, and take your chances!

The 3084 CPUs were numbered zero through three, but the 3090-400 are numbered one through four, and the 3090-800 probably will be numbered one through eight. Now, there are nine groups of CPU-specific variables, suffixed 0 through 8.

MVS 2.1.7 reversed the order in which Test Pending Interrupts (TPI) I/O is serviced. Formerly, TPI interrupts were serviced from low to high CPU number; now, the selection for handling TPI is from high to low CPU number. The TYPE70 I/O interrupt rate on 3090s is somewhat higher (5%?) than the TYPE78IO IOP activity rate. The difference may measure I/O redrive requests.

The recursive scan of the dispatcher queue before entry into the wait state, the primary cause of the low utilization effect discussed on page 98 in [1], has been significantly reduced. Instead of scanning all address spaces (including logically swapped ASIDs), the MVS 2.1.3 dispatcher scans only address spaces in memory. MVS 2.1.7 eliminates the recursive scan entirely, which has produced a measurable reduction in uncaptured CPU time. See CPUOVHTM in the RMFINTRV section of Chapter 40 in [2].

1.1 SERVICE UNITS PER SECOND

The Service Units per second value (SU_SEC, also called the Machine Dependent Constant, MDC) is now contained in the type 72 record. The table look-up in VMAC7072, which set SU_SEC from CPUTYPE and CPUVERSN, is not used after RMF 3.3, and updates to VMAC7072 with each new processor are no longer necessary. The importance of SU_SEC is unchanged, and its value can still be determined; see [1], pp 686-687.

Because the SU_SEC is derived by IBM from benchmarks (analyzed with you know who's software), the value has proven to be an excellent estimate of the processor capacity prior to the availability of actual hardware benchmarks for your own workload. A major source of confusion, however, is the apparent inconsistency in SU_SEC values between processors. For example, the SU_SEC for a 3081KX is 399, but a 3084QX value is 373.8. Since a 3084QX is simply two 3081KX's, why are they not equal? The reason is that the purpose of the MDC is to make CPU service units consistent.

When benchmarks show that a four-engine CEC records more seconds of task CPU time (type 30) than the same task recorded for a two-engine CEC, the MDC of the four-engine CEC must be smaller than the value for the two-engine CEC. The CPU time (TCB, SRB, and even uncaptured) is larger for a task in a dyadic than for one in a uni-processor. CPU time is accumulated while a task is executing, and if the task needs data that are not in cache storage, the task's CPU time continues to accumulate while the CPU competes with other CPUs for storage control. The competition (and therefore the measurable increase in CPU) grows with the number of engines. In actuality, the SU_SEC value is a single constant for an engine, and RMF de-rates the constant based on the number of engines currently on-line:

Table 1. Processor derate factor versus number CPUs

Processors on-line:	1	2	3	4	5	6
RMF 2.1.3 derate	100%	93%	88%	85%	80%	76%
Prior derate	100%	95%	92%	89%	-	-

An additional complication is the confusion of CPU speed with CPU capacity. In the preceding example, the 3081KX with a speed of 399 SU_SEC has two processors for a capacity of 798 SU_SEC, while the 3084QX speed of 373.8 provides a service unit capacity of 1495.2 per second. The following table provides both the speed and the capacity in raw service units for many IBM processors.

Table 2. CEC CPU "Speed" and "Capacity" values

CEC Model	MDC Speed	SU_SEC Capacity
155	42	42
168	151	151
3033U	261.3	261.3
4381-R14	200.6	401.2
3083JX	420	420
3081KX	399	798
3084QX	373.8	1495.2

CEC Model	MDC Speed	SU_SEC Capacity	CEC Model	MDC Speed	SU_SEC Capacity
3090-150	451.6	451.6	3090-150E	471.9	471.9
3090-180	750	750	3090-180E	823.5	823.5
3090-200	712.5	1425	3090-200E	765.9	1531.8
			3090-300E	724.7	2174.1
3090-400	667.5	2670	3090-400E	700	2800
			3090-600E	625.9	3755.4

As with any set of numbers, use these with caution. Recognize that they are aimed at maintaining consistency of service units for typical work and will not always apply to all processors or all workloads. Only an actual benchmark of your workload on your processor will provide exact measures.

1.2 ASID Count by State

These measures [1], pp 687-688, are good indicators of the overall system health, and they can specifically indicate constraints on the multiprogramming level (MPL) that can result from incorrect specification of MIN/MAX MPL values for each domain. Thanks to Bill Mullen's 1986 CMG presentation for the following guidelines:

READYAVG (the average number of IN and READY address spaces) should be less than 10% of INAVG. A too-high percentage of IN memory ASIDs is one result of specifying an MPL that is too high, which can cause significant queueing delays waiting for CPU dispatch. The suggested 10% value will maintain a queue (so the processor is not idle), but the delay will be small.

OUTAVG (the average number of OUT and READY address spaces) should not be greater than one for each active CPU engine. A higher value indicates a constraint on domain MIN/MAX values that could be examined by tracing (TYPE76) domain utilizations or by the display domain operator command.

LRDYAVG (the average number of LOGICALLY OUT and READY address spaces) should be zero. Like OUTAVG, it indicates possible domain constraints.

2. Vector Facility Measurement

The optional vector facility is described by its on-line status (VFONn) and the affinity duration (VFAFFTMn) for each CPU engine.

Vector facility affinity time is the duration of CPUTCBTM accumulated for all tasks having affinity to the vector processor. It is captured separately for each CPU engine. When a task first uses a vector instruction, it is given affinity to only the processors that have the vector facility. It keeps this affinity until the dispatcher detects that vector instructions have not been used for a long time. (A long time is 50 milliseconds of TCB with no vector instruction.) A task that has lost its vector affinity because of non-use can regain affinity when it again issues a vector instruction. Vector affinity time, VFAFFTMn, is accumulated for each engine that has a vector processor only when a task with vector affinity is dispatched and is executing under its TCB in that engine. Only one task can accumulate VFAFFTMn at one time in one engine, although several vector tasks can be waiting to be dispatched on that same engine.

Affinity time should be used for capacity planning and cost recovery of the vector processor. There is no more vector facility capacity when those engines with a vector processor record 100% affinity time. Because affinity time is a part of CPU TCB time, 100% affinity time occurs when VPAFFTMn equals CPUTCBn and the processor is 100% active. CPUTCBn is not directly measurable (except on a UP), but MVS uncaptured CPU time and SRB CPU time cause the measured VPAFFTM at 100% capacity to be significantly less than 100% of DURATM.

Vector affinity time is also accumulated by step in the TYPE30_4 and TYPE30_5 data set variables VPIAFFTM and VPUAFFTM. Two additional step level variables, VPIUSETM and VPUUSETM, describe the duration of actual vector processor execution within affinity time. See the TYPE30 section in Chapter Forty in this supplement.

For additional reference material on the vector facility, see Paul Dorn's excellent Washington Systems Center technical bulletin, "MVS/SP 2.1.3 Vector Facility Enhancement Overview," LG66-0227-00, and the "3090 Vector Facility Technical Reference," GG24-3058-0.

3. TYPE71

3.1 RMF Paging and Swapping Activity

The existence of extended memory (originally named expanded memory) has made significant changes in the TYPE71 data set, especially with swap counting and swap possibilities. With MVS 2.1.3 or later and on a 3090 processor with extended memory, Figure 1. describes the the possible swap states and their transitions.

The four boxes in Figure 1 are the four states in which a swappable task can be found. The six vectors indicate the possible transitions that a task can follow among these four states. The letters in parentheses and the phrases in quotes describe in which report column (see below) RMF prints these swap rates. The names in uppercase are the prefixes of the MXG variables that measure the transitions.

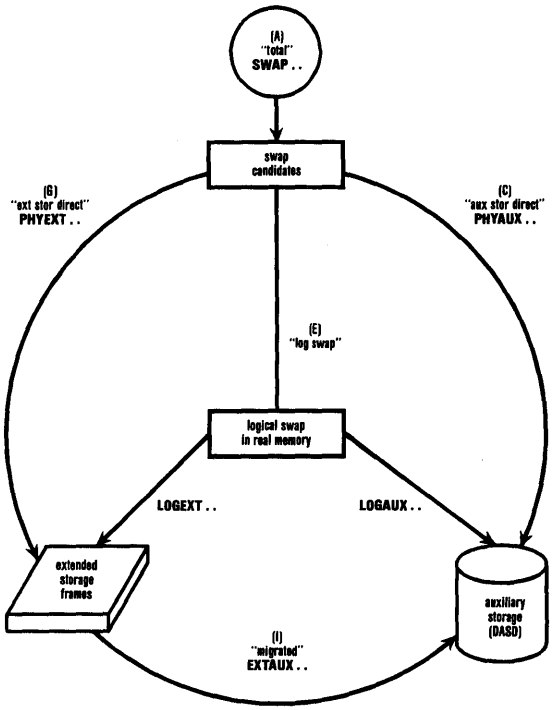


Figure 1. All Possible Swap States and Transitions

SWAP.. -- swap candidates count (rate per second)

Swap candidates will initially change state by one of the following three possible transitions:

PHYEXT.. -- physically swapped directly into extended storage

PHYAUX.. -- physically swapped directly to auxiliary (DASD) storage

logical -- logically swapped in real memory.

Tasks that are logically swapped in real memory (that do not become ready before the think time limit) could then undergo one of the following two possible transitions:

LOGEXT.. -- physically swapped from logical to extended storage

LOGAUX.. -- physically swapped from logical to auxiliary storage.

Tasks that have been moved to extended storage (that do not become ready before their extended storage space is needed) will go to only one state:

EXTAUX.. -- migrated from extended storage to auxiliary storage. (Pages migrated are moved to real storage and then moved to DASD.)

There are eleven reasons why a task may be swapped. These reasons are described on pages 715-717 in [1], by the SWAP.. variable names that end with the following swap reason code suffixes:

Table 3. Swap Reason Codes

blank - total for all eleven swap reasons
AS - auxiliary storage shortage (very unlikely)
DW - detected wait
EX - exchange on recommendation value
NQ - enqueue exchange
NS - transition to non-swappable (not many)
RS - real pageable storage shortage (very unlikely)
TI - terminal input wait
TO - terminal output wait
US - unilateral - SRM overcommitted or domain MPL exceeded
VR - request swap (not many)
WT - long wait.

These eleven reason codes (plus the total for all reasons) for each of the six possible transitions are stored in seventy-two swap rate (per second) variables.

The RMF swap placement activity report describes how SRM handles these swap requests and how effective SRM is in the way it satisfies these requests. Each row of the RMF report describes one swap reason. The following columns on the RMF report can be calculated directly from MXG variables:

Table 4. RMF Report Calculation Equations

Column and Heading on RMF Report	Calculated as
A total	= SWAP
B aux stor total	= PHYAUX + LOGAUX + EXTAUX
C aux stor direct	= PHYAUX
D aux stor via transition	= LOGAUX + EXTAUX
E log swap	= SWAP - (PHYEXT+PHYAUX)
F log swap effective	= SWAP - (PHYEXT+PHYAUX+ LOGEXT+LOGAUX)
G ext stor direct	= PHYEXT
H ext stor total	= LOGEXT + PHYEXT
I migrated	= EXTAUX
J ext stor effective	= LOGEXT + PHYEXT - EXTAUX
K log swap/ext stor eff.	= SWAP - (PHYAUX+LOGAUX+ EXTAUX)

Although all of these swap rate fields measure some facet of extended storage, the most important swap rate is EXTAUX, the rate of swapping migration. A low migration swap rate indicates that extended storage pages are not being moved to DASD, and the full benefits of extended storage are being achieved. (Although the EXTAUX transition is from extended to auxiliary storage, the actual physical pages first move to real storage, and are then written out to auxiliary DASD.)

The flow chart of Figure 2 was originally prepared by Dr. Jeff Buzen for the BGS User's Group meeting prior to CMG 86. It is presented here with appreciation and minor revisions. The chart follows the decision logic within MVS that determines which transits are actually taken. The letters identify the RMF report column.

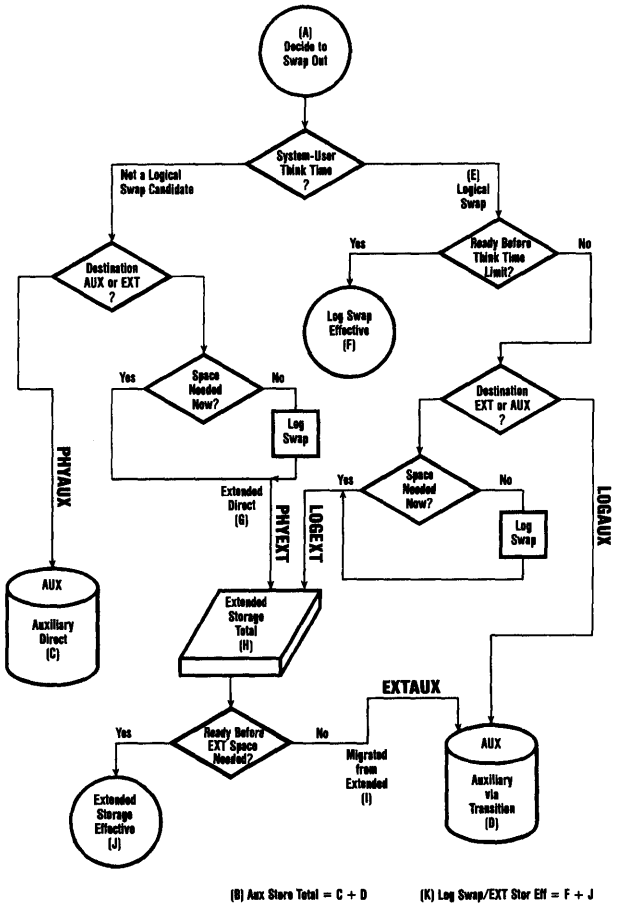


Figure 2. MVS/XA Extended Swap Decision Logic

The RMF paging activity report has changed somewhat as a result of the extended storage. The following figures identify which MXG variables are printed in which location on the RMF report.

Table 5. RMF Main Storage Paging Rates per Second

category	page reclaims non-swap	page in swap	page in non-swap	page out swap	page out non-swap
pageable system areas (non vio)					
lpa	LPARECLM		LPAGINS		
csa	PVTCAREC		PVTCAIN		PVTCAOUT
address spaces					
vio	PVTVAMR		PVTVAMI		PVTVAMO
non vio	PVTNPREC	PVTSPIN	PVTNPIN	PVTSPOUT	PVTNPOUT

All other columns on this RMF report are either sums or percentages of these thirteen values.

main storage frame counts number of samples NRSAMPLE

	min	max	avg
total frames			
lpa	LPAPGMN	LPAPGMX	LPAPGAV
csa	CSAPGMN	CSAPGMX	CSAPGAV
private	PRVPGMN	PRVPGMX	PRVPGAV
unused	PVTAFCMN	PVTAFCMX	PVTAFCAV
*total	FIXEDMN+ PAGBLMN	FIXEDMX+ PAGBLMX	FIXEDAV+ PAGBLAV
fixed frames			
sqa	SQAFXMN	SQAFXMX	SQAFXAV
lpa	LPAFXMN	LPAFXMX	LPAFXAV
**csa	CSLPFXMN	CSLPFXMX	CSLPFXAV
lsqa	LSQAFXMN	LSQAFXMX	LSQAFXAV
private area	PRVFXMN	PRVFXMX	PRVFXAV
***below 16 meg	FIXLOMN	FIXLOMX	FIXLOAV
total fixed	FIXEDMN	FIXEDMX	FIXEDAV

nucleus frames at end PVTFPFN

total online system frames PVTPOOL+PVTFPFN

Notes:

* This total is also equal to PVTPOOL.

** CSLPFX.. (not CSAFX.. as given in [1] page 706) is correct.

*** These frames below 16 meg are included in total fixed (which itself is the sum of SQA, LPA, CSA, LSQA and the private area.)

RMF Paging Activity Report - continued

real storage movement
total rate PVTMVTOT

rate of page movement
to es PGMVTOEX

migration rate PGMIEXAU

available esf
min AVLEXTMN
max AVLEXTMX
avg AVLEXTAV

	high uic	migr age
min	HIUICMN	MIGAGEMN
max	HIUICMX	MIGAGEMX
avg	HIUICAV	MIGAGEAV

esf configuration

installed	online
EXTFRMIN	EXTFRMON

local page data set slot counts

	min	max	avg
available slots	SLOTUNMN	SLOTUNMX	SLOTUNAV
vio slots	SLOTVIMN	SLOTVIMX	SLOTVIAV
non-vio slots	SLOTNVMN	SLOTNVMX	SLOTNVAV
bad slots	SLOTNGMN	SLOTNGMX	SLOTNGAV
total slots	SLOTLOMN	SLOTLOMX	ASMSLOTS

3.2 Extended Storage Paging Versus Swapping

The preceding discussions might lead you to believe that extended memory is only used by swapping, but that is not true. RSM, Real Storage Management, treats extended storage as an extension of real memory, and both swap and nonswap pages are moved from real storage to extended storage. By moving pages to extended storage instead of to auxiliary storage, a page-in can usually be avoided.

3.3 Control Mechanisms for Extended Storage

The Real Storage Manager (RSM) controls the utilization of extended storage by using a value called the criteria age to determine when a page should be sent to extended storage. The criteria age is specified by keywords in OPT for various types of pages and for three types of users. RSM then compares the selected criteria age value to the following two measures of real storage contention:

HIUIC:
System-high unreferenced interval count (UIC). This is the age in seconds that a page has remained unreferenced in real storage. A low value indicates contention for storage.

MIGAGE:
Migration age. This is the average age in seconds that pages have remained unreferenced in extended storage.

RSM wants to send to extended storage only those pages that are likely to be referenced again before being migrated to auxiliary storage. The decision logic depends on the type of page involved, and on the measures of HIUIC, MIGAGE, and think time:

Type of Page:	Sent to Extended Storage when:
Swap	HIUIC + MIGAGE > Criteria Age
Terminal Wait	HIUIC+MIGAGE-Think Time > Criteria
Stolen, Virtual Fetch, Page Out	MIGAGE > Criteria Age

The type of user is specified by the following keywords:

0=nonswappable, common, or privileged
1=all others not in type 0 or type 2
2=terminal wait swap, or TSO stolen or paged-out.

Based on both page type and user type, the criteria age is selected from the following table of defaults, which also summarizes the criteria test:

Table 6. Extended Storage Criteria Age

OPT Keyword	Page Type	Type of User Owning Page 0	1	2
ESCTPOC	Changed page-out pages	100	100	100
ESCTPOU	Unchanged page-out pages	100	100	100
ESCTSTC	Changed stolen pages	0	20	15
ESCTSTU	Unchanged stolen pages	0	20	15
ESCTSWTC	Changed swap-out pages	100	100	60
ESCTSWTU	Unchanged swap-out pages	100	100	60
ESCTSWWS	WKSET pages ready for swapout	100	100	50
"	Terminal swap pages		100	
ESCTVF	Virtual fetch pages		15	

Page Type	Criteria
Changed page-out pages	MA > CA
Unchanged page-out pages	MA > CA
Changed stolen pages	MA > CA
Unchanged stolen pages	MA > CA
Changed swap-out pages	UIC+MA > CA
Unchanged swap-out pages	UIC+MA > CA
WKSET pages ready for swap-out	UIC+MA > CA
Terminal swap pages	UIC+MA-TT > CA
Virtual fetch pages	MA > CA

3.4 Sequence of Swap Actions

When an address space that was swapped directly from real storage to auxiliary storage (PHYAUX and LOGAUX swaps) is to be swapped back in, ASM reads and writes these working set pages from DASD to real memory. This is called a single-stage swap-in.

If an address space is swapped from real storage to extended storage, RSM and SRM split the address space into primary and secondary working sets. The primary working set is the LSQA pages, fixed pages, and the first page from each virtual storage segment. (The first pages are necessary for reconstruction of the virtual segment and page tables at swap-in.) RSM manages the primary working set as one entity. The secondary working set is all other working set pages.

If the address space is later selected for migration to auxiliary storage, RSM migrates the secondary working set in groups, based on the need to replenish the extended-storage-available frame queue. ASM then writes these secondary working set pages as group requests to the page data sets. After the entire secondary working set has been selected for migration, the RSM swaps the primary working set from extended storage into real storage and then writes it out to DASD auxiliary storage.

Once an address space has been split, the RSM must return the entire primary working set to real memory before the address can become dispatchable. If the address space has been migrated, the primary working set is swapped first, and then the secondary working set is swapped. This is called a two-stage swap-in.

3.5 Extended Storage Measurements

Given the preceding discussion, the actual measurement of extended storage is simple. The most important measure of utilization is PGMIEXAU, the actual rate of page migration from extended storage to auxiliary storage. If there is no movement of pages from extended to auxiliary, then extended storage is clearly not overcommitted. To place page movement in perspective, however, PGMIEXAU must be compared (perhaps as a percentage) to PGMVTOEX, the total page movement rate

into extended storage. Note, however, that it is not possible to separate page movement or migration due to swap pages from page movement or migration due to nonswap pages. PGMVTOEX and PGMIEXAU disregard the type of page being moved when they count page movement.

In current 3090 architecture, pages to be migrated are actually moved first from extended storage to real storage, and then they are physically moved from real to auxiliary storage; but that design is likely to change.

If appreciable migration from extended to auxiliary storage occurs, only then is it worthwhile to determine which swap reasons are involved by examining the eleven EXTAUX.. variables. Note that these variables count the rate that address spaces are moved or migrated, rather than counting the rate that pages are being moved.

4. TYPE72 WORKLOAD MEASUREMENT

4.1 RMF Workload Activity

The SU_SEC value, page 686 in [1], is no longer acquired by table look-up from CPUTYPE and CPUVERSN in TYPE70. RMF 3.3 added the value of SU_SEC in the type 72 record. SU_SEC is of critical importance because it is used to convert CPU service units into CPU seconds in TYPE72. By then subtracting the TYPE72 control performance group CPU times from the TYPE70 CPU active time, the MVS uncaptured CPU time can be measured. This is done in the RMFINTRV data set, which is described in Chapter Forty in [2]. SU_SEC also identifies the speed of the processor engine, which is important for capacity planning, and permits normalization of processor measurement for machines of different speed. See the new section on service unit measurement in Chapter Twenty-Six in [2].

4.2 Domains, Performance Groups, and Periods

TYPE72 observations are created for each period of each performance group. A performance group is simply a number that is assigned to a task by the installation for resource reporting and labeling. A task keeps the same performance group for its entire execution duration (except that batch steps can change performance groups by specifying a different value of PERFORM= on an EXEC statement). The SYS1.PARMLIB IEAICSnn member (which ensures that work is controlled) defines the performance group for each task in a hierarchy that you create.

Performance groups are assigned to tasks in ICS first by subsystem:

```
STC
JES2/JES3          PGN=nn
TSO
```

unless you specify performance group by job name or job class:

```
TRXNAME=MYJOB      PGN=mm
TRXCLASS=A         PGN=oo
```

or unless you specify performance group by userid:

```
USERID=BERNIE      PGN=pp
```

Performance groups can be subdivided into periods so that the resources consumed by transactions or steps can be controlled. You may wish to reward the interactive users by giving them high levels of service, but you may also wish to punish the batch user who executes his stage-one SYSGEN under TSO. The periods are defined by the DUR parameter, which sets the number of service units to be permitted in each period. When a task exceeds the DUR value, the SRM detects that event and transfers the task to the next period:

Period One	Period Two	Period Three
DUR=300	DUR=1000	
PRTY=HI	PRTY=MED	PRTY=LOW
DMN=5	DMN=4	DMN=6

Figure 3. Periods within a Performance Group

The performance group period defines the domain in which the task is to execute, and the domain definitions determine the priority and service level that the task receives.

Domains are numbered collections of tasks that are to be treated equally with regard to service-unit consumption. Note that tasks commonly change their domain during a transaction's execution.

Because the SRM chooses domains for service delivery, a large number of domains is unnecessary, unwise, and can increase the MVS overhead. Performance groups have no overhead; they are simply buckets in which the service units and transaction counts are stored when a transaction ends. Therefore, it is common, and in fact wise, to have a large number of performance groups. It is strongly suggested that you place each long-running address space (that is, each CICS, each IMS control region, each IMS message region, the RMF, and so forth) in a separate control performance group. This permits TYPE72 data to be invaluable in determining workload growth over time as well as measuring the percentage of the peak-hour resources that each workload consumes. Although separate performance groups for each long-running task are desired, all tasks that are alike (for example, IMS message regions) can be mapped to the same domain so that the SRM will treat them equally.

The ratio of RESIDTM to DURATM gives the average number of address spaces in this performance-group period (that is, in this domain) that were resident in real memory. This is commonly referred to as the multiprogramming level (MPL) for the performance-group period. For example, the ratio can be calculated for all periods of TSO performance groups to determine the average number of TSO users in real memory.

The difference between ACTIVETM and RESIDTM is the swap-in delay time. If this difference is divided by TRANS, the average swap delay (in seconds) for each transaction is known.

5. TYPE74 DEVICE MEASUREMENT

5.1 RMF Device Activity

In addition to the following discussion of device measurement at the device level, see the TYPE78IO, TYPE78CF, and TYPE78CU sections in [2].

TYPE74 is the primary source of information for managing and measuring the DASD I/O environment in MVS/XA. Although TYPE74 data can also be used for some measurements of tape I/O activity, the techniques described in Chapter Fourteen are generally more appropriate for tape analysis.

MVS/XA captures data that are recorded in hardware monitors that are built into the channels. The data measured at the device level provide complete and

detailed data on the I/O activity. The actual data that are recorded in type 74 records consist of a number of duration measurements. For example, the duration during which data were actually being transferred between the CPU and the device is measured by DEVCONTM (the number of seconds of device connect time during the RMF interval).

In addition to this absolute duration measure, you can examine each measure from two other useful perspectives. If you divide the measured duration by the RMF interval duration, you get the percentage of time when data transfer was occurring (PCTDVCON). If you divide the measured duration by the number of I/O operations, you get the average milliseconds of connect time (AVGCONMS) for each SIO (actually, for each start subchannel (SSCH) instruction). Because all three perspectives are useful, MXG creates a variable for each perspective in TYPE74.

The power of TYPE74 analysis lies in the completeness of the measures of both the utilization and the delays at the device level. The next figure identifies the major durations (and their alternate perspective variables) that are created in TYPE74.

```
      AVGACTMS  - Average Active Milliseconds per SSCH
      DEVACTTM  - Total seconds active time
      PCTDVACT  - Percent of interval when active

|-------------------------------------------------------|
                          Active

        AVGPNDMS              AVGDISMS      AVGCONMS
        DEVPNDTM              DEVDISTM      DEVCONTM
        PCTDVPND              PCTDVDIS      PCTDVCON

|-----------------------|-----------|-------------|
        Pending           Disconnected  Connected

  AVGPNCHA AVGPNCUB AVGPNDEV
  PCTPNOTH PCTPNCUB PCTPNDEV
|--------|--------|--------|-----------|-------------|
   Pend     Pend     Pend
    for      for      for
  Channel  Control  Device
   Path      Unit     Busy
   Busy      Busy

|-----------------------|-----------|-------------|
     Pending              Disconnected  Connected

Waiting for available    Device in use      Actual
channel path, control    but not actually   Data
unit, or device busy     transferring data: Transfer
                          SEEK (arm motion)
Only 309x CEC has all     SET SECTOR (RPS)
three components of       RECONNECT (RPS)
pending time measured.

  Typically 1-5 ms       Typically 9-27 ms   Min. value
   per SSCH               per SSCH            1.3 ms  4K
                                             14.9 ms 47K
```

Figure 4. MVS/XA Device Measurement Components

In addition to the actual hardware measurements of disconnected and connected durations, RMF samples UCB bits to calculate the additional percentage of time when the device was reserved but was not involved in an I/O operation (PCTDELAY). The sum of PCTDVDIS, PCTDVCON, and PCTDELAY is PCTDVUSE (the percentage of time that the device was in use). PCTDVUSE describes how long the device was "tied-up" and could not be used to service a request from any other system.

The MVS/370 measure of device utilization, DEVBUSY, was replaced by PCTDVUSE:

```
                          PCTDVUSE
  |-----------------------------------------------|
                    Device Utilized

       AVGDISMS       AVGCONMS
       DEVDISTM       DEVCONTM
       PCTDVDIS       PCTDVCON       PCTDELAY

  |---------------|--------------|-----------------|
    Disconnected     Connected      Reserved with
                                      no I/O
```

Figure 5. MVS/XA Device Utilization Components

Yet another measure is sampled by RMF: the average number of I/Os that were enqueued in IOS (that is, delayed in the input/output supervisor because an I/O was already outstanding to this device from this system). Knowing the average number of I/Os that were queued permits the calculation of the average milliseconds of IOS queue time for each SIO (AVGIOQMS). With this additional value, the average response time in milliseconds for each SIO can be determined:

```
                          AVGRSPMS
|-------------------------------------------------------|
        Average Response Milliseconds per SIO

  AVGIOSMS      AVGPNDMS        AVGDISMS      AVGCONMS
                DEVPNDTM        DEVDISTM      DEVCONTM
                PCTDVPND        PCTDVDIS      PCTDVCON

|----------|--------------|---------------|-------------|
 IOS Queue     Pending        Disconnected    Connected
```

Figure 6. Average DASD Response Components

5.2 RMF Device Report

The RMF direct access device activity report column headings are shown below in lowercase; the MXG variable directly corresponding to each column is indicated in uppercase. Using PROC PRINT SPLIT='*'; with a VARIABLES statement naming these variables would create the report.

Table 7. RMF Device Report Variables

dev num	volume serial	lcu	device activity rate	avg resp time	avg iosq time	avg cub delay
DEVNR	VOLSER	LCU	IORATE	AVGRSPMS	AVGIOMS	AVGPNCUB

avg db delay	avg pend time	avg disc time	avg conn time	% dev conn	% dev util
AVGPNDEV	AVGPNDMS	AVGDISMS	AVGCONMS	PCTDVCON	PCTDVUSE

% dev resv	avg ds open	% alloc	% mt pend
PCTRESVD	AVDSOPEN	PCTALOC	MOUNT

6. TYPE78.. I/O FOR 3090

Seven different TYPE78.. data sets are created from the three subtypes of the type 78 RMF records. Subtype 2 contains virtual storage measures that create TYPE78PA, TYPE78SP, and TYPE78VS data sets. The subtypes 1 and 3 contain I/O measures that are structurally different depending on the processor on which the data are created:

Table 8. Sources of I/O Measurement

308x and 4381		309x	
MXG Data Set	Subtype	MXG Data Set	Subtype
TYPE78	1	TYPE78CF	3
TYPE78CF	1	TYPE78CU	3
		TYPE78IO	3

Primary I/O Measurement Sources -- 308x Processors

Data Set	Variables
TYPE73	CHAN PCHANBY
TYPE78CF	LCUID CHPID CU1-CU4
TYPE78	LCUID IORATE AVGENQUE PCTALLBY PCTDEFER PCTDEFDV PCTDEFCU
TYPE74	LCU DEVNR VOLSER IORATE PCTDVPND PCTDVDIS PCTDVCON AVGRSPMS

Primary I/O Measurement Sources -- 309x Processors

Data Set	Variables
TYPE78IO	IOPIQID IOPACTRT AVGIOPQL
TYPE78CF	LCUID CHPID CU1-CU4 CHPIDTKN PCTPTHBY PCTCUBSY
TYPE78CU	LCUID CUHACTRT AVGCUHQL
TYPE74	LCU DEVNR VOLSER IORATE PCTDVPND PCTDVDIS PCTDVCON AVGRSPMS

6.1 RMF I/O Queueing Activity

The RMF device activity report has already been described in the preceding TYPE74 section in this supplement. The RMF I/O queueing activity report described below combines information from the TYPE78IO, TYPE78CU, and TYPE78CF data sets, which are described in more detail in their own sections in [2]. The following report is described as it exists for RMF 3.4.1 on a 309x processor:

----------from TYPE78IO----------

iop	activity rate	avg q lngth
IOPIQID	IOPACTRT	AVGIOPQL

-----from TYPE78CU-----

lcu	contention rate	delay q lngth
LCUID	CUHACTRT	AVGCUHQL

----------from TYPE78CF----------

ch path busy	control units 1	2	3	4	chan path	chpid taken	% cu busy
Note 1	CU1	CU2	CU3	CU4	CHPID	CHPIDTKN	PCTCUBSY

Note 1:
For each TYPE78CF observation, the PCTPTHBY variable exists, and it is identical with the TYPE73 PCHANBY variable for that CHPID. The RMF queueing activity report takes the product of PCTPTHBY for each CHPID within each LCUID as the estimated probability that all channel paths are busy within this LCUID.

Figure 7. RMF I/O Queueing Activity

There is one observation for each LCUID in TYPE78CU. In TYPE78CF, there is one observation for each CHPID within each LCUID, with up to four CUs connected to that CHPID. Thus, there may be several lines from TYPE78CF for each LCUID from TYPE78CU.

6.2 TYPE78CF MVS/XA Device Configuration

There is one observation in TYPE78CF for each combination of a CHPID (Channel Path ID) within each LCUID (Logical Control Unit ID).

TYPE78CF is built from type 78 subtype 1 records if the processor is a 308x or 4381, or from the type 78 subtype 3 record for 309x processors.

Prior to 309x processors, the TYPE78CF data set was only a static description of the physical connectivity of physical control units, their channel paths, and their logical control units. The 309x machines have added four important utilization statistics: the percentage that all channel paths were busy, the rate at which I/Os were taken on this CHPID for this LCUID, the percentage that all paths were busy to this LCUID, and the percentage that physical control units were busy on this CHPID LCUID combination. While TYPE74 still remains the primary source of I/O delay measurement, once a candidate problem device has been identified, TYPE78CF can provide amplification of potential path problems.

6.3 TYPE78CU MVS/XA 309x Control Unit Header Queue

There is one observation in TYPE78CU for each LCUID (Logical Control Unit ID). TYPE78CU is built from the type 78 subtype 3 record for 309x processors only. It describes the activity and queueing for the control unit header queue.

6.4 TYPE78IO MVS/XA 309x Input-Output Processor Init Que

There is one observation in TYPE78IO for each IOP. TYPE78IO is built from the type 78 subtype 3 record for 309x processors only. It describes the activity and queueing for the IOP initiative queue.

7. TYPE78VS MVS/XA Virtual Storage Statistics

Some virtual storage measures in TYPE78VS are contained in sets of ten variables whose suffix (0-9) identifies the content of the variable. Some virtual storage measures have a suffix of 0-4 because that area exists only below the 16MB line. It is the area below 16MB that is most critical because many system functions can exist only below the 16MB line.

Table 9. Suffix of Virtual Storage Measures

--------Below 16MB--------					--------Above 16MB---------				
min value	time of min	max value	time of max	avg value	min value	time of min	max value	time of max	avg value
0	1	2	3	4	5	6	7	8	9

The time stamps (suffixes 1, 3, 6, and 8) are correct in the type 78 record (and, thus, in MXG), but a close examination of the RMF report will show time stamps that are earlier than the beginning of the RMF interval (even with RMF 3.4.1). The error results from truncation in an internal calculation in the IBM RMF report program. The RMF printed value is typically 1 to 4 seconds earlier than the correct value. This error has been reported to IBM.

The RMF virtual storage activity report from TYPE78VS data is shown below. Report columns are in lowercase, and MXG names are in uppercase.

```
                static storage map
 area          address     size

epvt          PRVTADHI   PRVTSZHI
ecsa           CSAADHI    CSASZHI
emlpa         MLPAADHI   MLPASZHI
eflpa         FLPAADHI   FLPASZHI
eplpa         PLPAADHI   PLPASZHI
esqa           SQAADHI    SQASZHI
enuc          NUCLADHI   NUCLSZHI
  -----16 meg boundary-----
nucleus
sqa            SQAADLO    SQASZLO
plpa          PLPAADLO   PLPASZLO
flpa          FLPAADLO   FLPASZLO
mlpa          MLPAADLO   MLPASZLO
csa            CSAADLO    CSASZLO
private       PRVTADLO   PRVTSZLO
psa                zero  PRVTADLO
```

```
                    allocated csa/sqa

                ---below 16M---  ---above 16M---
                min  max  avg   min  max  avg
sqa          SQAUSED,0-4       SQAUSED,5-9
csa          CSAUSED,0-4       CSAUSED,5-9

 allocated csa by key
  0          CSAUSE0,0-4       CSAUSE0,5-9
  1          CSAUSE1,0-4       CSAUSE1,5-9
  2          CSAUSE2,0-4       CSAUSE2,5-9
  3          CSAUSE3,0-4       CSAUSE3,5-9
  4          CSAUSE4,0-4       CSAUSE4,5-9
  5          CSAUSE5,0-4       CSAUSE5,5-9
  6          CSAUSE6,0-4       CSAUSE6,5-9
  7          CSAUSE7,0-4       CSAUSE7,5-9
  8-F        CSAUSE8,0-4       CSAUSE8,5-9

 sqa expansion into csa
             SQAEXPN,0-4       SQAEXPN,5-9

 plpa intermodule space  IMPLSPLO in plpa and
                         IMPLSPHI in eplpa
 plpa space redundant with mlpa/flpa PSRMSPLO in plpa
                                     PSRMSPHI in eplpa

                          ---below 16M---  ---above 16M---
                          min  max  avg   min  max  avg
csa
 free pages (bytes)          CSAFREE,0-4       CSAFREE,5-9
 largest free block          CSALARG,0-4       CSALARG,5-9
 allocated area size         CSAALOC,5-9       CSAALOC,5-9
sqa
 free pages (bytes)          SQAFREE,0-4       SQAFREE,5-9
 largest free block          SQALARG,0-4       SQALARG,5-9
 allocated area size         SQAALOC,0-4       SQAALOC,5-9

  maximum possible user region -- USERSPLO below and
                                  USERSPHI above
```

Figure 8. RMF Virtual Storage Report

8. MVS 2.2 ANNOUNCEMENT

MVS/XA 2.2.0 (JES2) and 2.2.1 (JES3) and associated RMF 3.5 have been announced for 3rd quarter 1987 availability. This discussion is based only on the IBM Announcement of those products.

8.1 NEW SMF RECORDS

36	ICF (Integrated Catalog Facility) Export/Import
41	DIV (Data In Virtual) Access
90.17	Subtype for Set PFK

8.2 NEW SMF DATA

The type 30 record will contain the number of Data In Virtual (DIV) rereads in the step and job termination records and in the interval records.

The type 72 performance group period data will contain (finally!) paging statistics for each period of each performance group.

If the type 4 and 34 SMF records are not already dead, MVS 2.2 is their final poison. The Task I/O Table (TIOT) is expanded from 32KB to 64KB in MVS 2.2, which allows for 3273 DDs (compared with only 1635 DDs). The type 4 and 34 SMF records do not support more than 1635 DDs, but instead contain a "not here" bit, indicating that the step had more than 1635 DDs and that I/O counts for the excess will only be found in the type 30 records.

The Selectable Unit Installed bit string in the type 70 record does not exist in MVS 2.2; it had not been updated since MVS/370.

8.3 SMF RECORD SUBTYPE SELECTIVITY

A bit in record types 30, 32, and 70-79 allow selectivity of record subtypes. The selectitity can be specified in SYS1.PARMLIB (TYPE,SUBTYPE) to selectively create records, or it can be specified in parameters to the IFASMFDP program for dumping selectivity by subtype.

8.4 INCOMPATIBLE CHANGES

IBM has stated that programs that do not use the relocatable format information in the type 30, 72, and 79 will be affected by incompatible changes to those records.

The documentation of these new changes in SMF data were released in April, 1987, in GC28-1411 MVS 2.2 Conversion Notebook Volume 2. IBM should be congratulated for the early release of the documentation of these changes which affect the format of existing data records.

8.5 MVS 2.2 PERFORMANCE ITEMS

The following items in the announcement should be noted.

Many control blocks were moved above 16MB to ease virtual storage constraints. This will likely affect all system monitors; check with your monitor vendor for a new version/release of their product.

Global Resource Sharing (GRS) will have a larger RSA (up to 32K, or 8 pages), the QWB will be variable length, and the SMPL is compressed from 96 to 28 bytes. This leads to more QWBs per page and to more QWBs per RSA, which should reduce delays in the GRS processing time.

The DUAL option for the JES2 checkpoint data set should reduce I/O on the SPOOL.

Multiple JES2 output PCEs can be defined, leading to improved print throughput.

All SVCs are now defined in IEASVCxx and the SVC table is now page protected. This requires all SVC users to use SVCUPDTE which will also affect system monitor vendors.

Expiration dates through year 2155 are now supported, with 99365 and 99366 protected from ever expiring. Why through 2155 you ask? Because there are 255 years from 1900 to 2155 and a one byte counter for the years since 1900 are used! SAS-provided formats and functions like SMFSTAMP8. and DATEJUL() will undoubtedly be revised to handle the new format. At present, however, only the EXPDT in the type 14 and 15 records, a packed decimal date without associated time, can be in the next millenium.

Now that extended storage has virtually eliminated the need for page or swap data sets, MVS 2.2 will support up to 256 separate swap or page data sets.

Up to 9999 report performance groups can be defined, with the limit of 999 control performance groups unchanged.

The RMF Monitor II value of CPU=101, which indicates that the CPU is actually 100% busy and in addition there was at least one ready task which was not dispatched during the last five elapsed seconds, will now count the actual number of undispatched ready tasks, up to a maximum of 28. You might even see CPU=128 in the future.

The Logrec buffer has been increased from 4KB to 16KB.

Partioned Data Sets and QSAM disk files can be concatenated in any order. It is no longer necessary to specify the largest blocksize first. Apparently, IBM must have finally found its own documentation of the SILI bit fondly remembered by OS/360 system programmers.

9. ONE MIPS AND TWO MEGS - SOME FOOD FOR THOUGHT

In the late 1970's, Walt Doherty, of IBM's T.J. Watson Research Center, concluded from studies of highly interactive development CMS users that each such user truly required only 1-2 MIPS and 2-4 MEGs, "on demand".

Studies of TSO user data suggests that no more than 10% of the logged on TSO users are actually active, that is, in and ready, or "on demand."

With 500 logged on TSO users on a 3090-400 with 128MB memory, only 50 users would be expected to be in and ready on this 50 MIPS machine, and each "on demand" user would have 2 MB of memory even if we allowed the operating system to own 28MB. Thus this machine would provide one MIPS and two MEGS to each "on demand" user.

Conversations with many sites with this hardware and workload seem to subjectively confirm that 1 MIPS and 2 MEGS provides unconstrained TSO response.

Another 3090-400 with 50MIPS and 128MB supported eight CICS applications which showed a total peak maximum active tasks (equivalent to the TSO in and ready, or "on demand") of 48. No response delays were noted, and the eight CICS applications recorded not one single page fault for the entire week. One MIPS and two MEGS seems also to meet CICS needs.

The advent of extended storage has finally provided our programs with enough real memory to execute, unconstrained by storage limitations. This in turn has reduced some of the operating system overhead which was consumed in serving the paging, but more importantly, the massive reduction in page service time (70 microseconds for extended memory, 30,000 micro seconds from disk) has eliminated significant delay to the user response. We see reduced resources for better response.

I believe that when the user notices the response time, it is already too long.

What will happen when you have the hardware to provide one MIPS and two MEGS for each "on demand" user? Your capacity planning headaches will go away, and computer growth will be tied directly to user population. The user population is constrained by new employment, which itself is constrained by profit. Rare is the company whose profit increases more than 10% per year.

Once you have provided unconstrained response to your users, only adding more users will require more hardware. Since you have already computerized almost all of the repetitive business functions, new development may add function and ease of use, but will not likely increase the number of concurrent business users of those repetitive business functions.

Ah, but what about those fourth generation languages that eat resources? Won't they fill my CPU and memory? There are certainly some industries which can take advantage of computer power in relational data bases to further the business purpose. I suspect, however, these companies are few and far between. Furthermore, how many employees will need to execute these long running complex transactions, and how many times a month will these "analyses" be required? If they produce complex results, is it not likely that many hours will be

required by humans to analyze the results, to establish marketing plans based on the results, etc.?

I am not suggesting that all computer growth is coming to an end. There are many well known problems which are just being feasibly solved which need much more power to be accurately solved. (Predicting weather has been described as needing 9 dimensions, while current computers can only support 3 dimensions, and then only for a 5-day prediction). But for the run-of-the-mill business data processing (invoicing, accounts receivable, returns, credits, payroll, etc., etc.), which has been the main driving factor of computer growth since the first 650 was shipped, I believe the compound growth rates we have seen are a thing of the past. If you install the right amount of hardware and software to do the job, then you won't need a job position to figure out how much hardware and software you need.

References

[1] Merrill, H. W., Merrill's Expanded Guide To Computer Performance Evaluation Using the SAS System, SAS Institute, Cary, N.C., 1984, 866 pages.

[2] Merrill, H. W., Merrill's Expanded Guide Supplement, SAS Institute, Cary, N.C., 1987, 624 pages.

MXG is a Registered Trademark of Merrill Consultants, Dallas, Texas.

SAS is a Registered Trademark of SAS Institute, Cary, N.C.

EXPERT SYSTEMS AND MVS PERFORMANCE MANAGEMENT

J. van Kinsbergen
Boole and Babbage, Inc.

ABSTRACT

This paper will focus on the author's definition of expert systems, and its application to and issues with the problems of MVS performance management. It will establish what is and what is not expert system technology and then the time dimension of performance management. Finally, interfacing expert systems to the MVS environment will be discussed.

THE PROMISE OF EXPERT SYSTEMS:

PRODUCTS DEVELOPED WITH EXPERT SYSTEM TECHNOLOGY WILL PROVIDE LABOR-SAVING DEVICES THAT WILL ALLOW SYSTEMS PROGRAMMING EXPERTS TO FOCUS ON REAL ISSUES AND OPPORTUNITIES WORTHY OF THEIR EXPERTISE. EXPERT SYSTEMS TECHNOLOGY WILL NOT REPLACE EXPERTS, BUT WILL ENHANCE THEIR ABILITY TO FOCUS, AND THE NOVICE'S ABILITY TO PERFORM AS AN EXPERT.

1. WHAT IS EXPERT SYSTEM TECHNOLOGY?

JUST BECAUSE A SOFTWARE IMPLEMENTATION DOES EXPERT THINGS DOES NOT MAKE IT AN EXPERT SYSTEM.

It is instructive to discuss first what expert system technology is not. Expert system technology does not merely:

1. Do expert things
2. Mimic human reasoning
3. Capture knowledge
4. Have rules

While expert system tehcnology applications in many cases will demonstrate some or all of these attributes, the same can be said for many other software systems. Just merely doing the above does not constitute an expert system; as we well know, there have been many implementations over the last thirty years of software packages that could say they accomplished the above items.

EXPERT SYSTEM TECHNOLOGY IS THE FIFTH GENERATION IN THE EVOLUTION OF PROGRAMMING LANGUAGES (I.E., COMPILERS)

In my opinion expert system technology is the current generation of the evolution of application development languages over the last thirty-five years. The generations are as follows:

1. First generation, machine languages
2. Second generation, assembler languages
3. Third generation, procedura languages, e.g., PL/1, COBOL
4. Fourth generation, form-driven "programmerless" languages
5. Fifth generation, expert system languages

Another part of the confusion comes from the terminology that has been created. There has been extraordinary work done over the last several years in the development of expert system technology; however, I believe that it has been allowed into the commercial marketplace without proper "scrubbing" by the general marketing and business community. We therefore have new terms to define old activities. At this point I would like to define these equivalences.

MUCH OF THE TERMINOLOGY ASSOCIATED WITH EXPERT SYSTEMS (INCLUDING THEIR VERY NAME) CAN BE CONSIDERED TO BE REDEFINITIONS OF HISTORICAL TERMINOLOGY. THE MOST IMPORTANT ADVANCEMENT IN EXPERT TECHNOLOGY THEN IS THE INFERENCE ENGINE OR APPLICATION CONTROL PROGRAM.

Fifth Generation Language

Expert System Technology
Knowledge Base
Inference Engine
Rules
Knowledge Engineer

Third Generation Language

Compiler with Run Time Option
Source Code
Run Time Option
Application Logic
Application Analyst

For example, the knowledge engineer is the person who sits down with the user expert and lays out what is required for the implementation of the application. With new expert systems technology the applications are more varied and complex; nevertheless, the knowledge engineer can be considered to be the application analyst of the future.

The knowledge base consists of rules implemented by the "knowledge engineer" in a form consistent with what we have historically called source code.

The most significant new item is the inference engine. The inference engine can be looked at as an extension of the operating system control program. As you are well aware, operating systems provide two major functions:

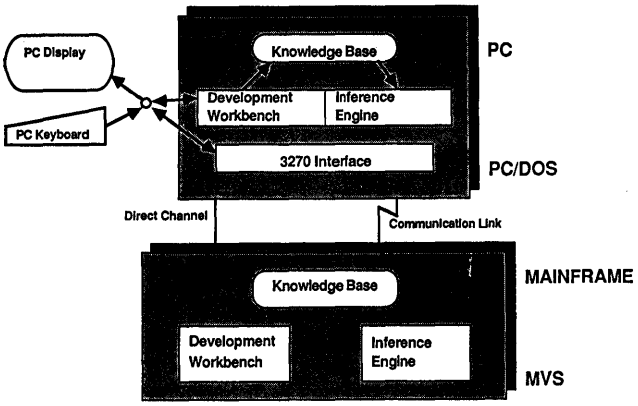
1. Control the interaction of work units in the system
2. Provide common services

The inference engine does exactly this in concert with the compiled source code created by the knowledge engineer. The inference engine is an application control program that runs in the application address space; controls the flow of work (rules), and provides common services to tasks within this application. In my opinion it is an extraordinary development in the evolution of programming languages and frees the programmer analyst from writing code that controls the flow of work within the application. That is, he does not have to write code that decides when to set or test a flag, or read input, etc. I am sure that you will agree with me that a major portion of application development consists of creating code that just gets the application from one piece of logic to another. The inference engine component of an expert system provides this capability. The robustness of the expert system can be measured by the capabilities of the inference engine.

The knowledge base itself consists of separate knowledge objects that are managed by the inference engine. These knowledge objects can consist of rules, variables, messages, reports, external files and/or external processes. The existence of the inference engine, and the technology that supports it, allows the developer to treat these knowledge objects as separate items in the development process. "If - then - else" rules can be entered into the system in an independent and "asynchronous" manner. The inference engine will tie these rules to existing or future variables as well as input and output. This inferencing process, both on the development side and the execution side, represents the most significant new technology provided by expert system language capability.

Figures one and two represent an example of an expert system consisting of two parts: a development system and the execution system, both of which execute in both the MVS mainframe and PC/DOS environments. This allows the developer to take advantage of the significant productivity gains in the PC environment but create code that is targeted for the mainframe environment. There is complete portability between the two, and thus significant productivity gains by taking advantage of the best of both worlds.

FIG 1. PC/MAINFRAME ARCHITECTURE



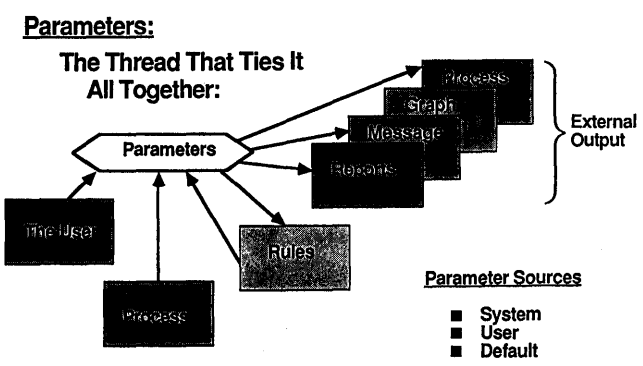
The development system has several "object editors" that allow creation and editing of the following objects:

1. States. This is the major modular unit of an application, equivalant of a subroutine.

2. Types. This unit allows the developer to create his own record formats or to use the types that come with the system. The built-in types include real and integer numbers as well as strings, lists, sets, and booleans.

3. Rules. Are of the production rule form: if - then - else. Major capabilities exist that allow iterative rules as well as the equivalent of case statements allowing multiple logic to be included in a single rule.

FIG 2. TECHNOLOGY EXAMPLE



4. Input and output and logic modules. Managed through the state, report, message, graph, and process objects. Each of these objects has input rules associated with them which control when the inference engine will fire off the function. And finally the process object allows access to external data and/or "foreign code". Again these process objects can be invoked based on input selection, and rules as described above.

The inference engine provides both forward chaining and backward chaining capabilities. Most effective in my opinion is the backward chaining process. In this case we establish a goal to be resolved, and the inference engine will backward chain through the logic to support it. An example would be the creation of a report with specific output variables imbedded. These variables are defined to the system in the form of parameters. The inference engine will start with these parameters and search its rule and process base in order to determine how the parameters are satisfied. In some cases, access to the keyboard will occur. As the inference engine determines a rule that can satisfy a parameter, it will in most cases require other parameters in order to fire the rule. This will require chaining further back to other rules, processes, or the keyboard again. This process continues until the very lowest level parameters are determined, which then cause the rules, etc., to be escalated back until the parameters in the goal (report) have been satisfied. This allows tremendous nested logic to be created in a manner that is completely beyond the mental capacity of a procedural approach. Because the inference engine controls the flow of work, it also can provide significant other run time facilities to support the application. This includes explain facilities as well as run time debugging options that enhance the understanding and maintainability of the application.

This technology example provides the following attractive features:

1. Significantly increased development productivity due to the object oriented approach to development as well as the basic workbench capabilities.

2. Because of the independence of the rules, the limitations associated with the nested approach to logic statements disappear. Logic can be pursued that heretofore was impossible.

2. MVS INTERFACE ISSUES

Expert systems operate best when they are digesting "facts". They are not designed just to deal with presentation and formatting of numerical data as are previous generations of development languages. Their focus is to take facts; draw conclusions, and allow the creation of an analytical set of logic. Procedural languages on the other hand are much better for numerical processing. However, in any production data processing environment numerical data obviously is a key component, in particular, in the performance management area. We obviously have been generating extraordinary amounts of data and information for years and not too many facts and conclusions. The interface of expert systems to the MVS environment for the inplementation of performance management products will be eased if the data that is presented to the expert system is presented in the form of facts. An example would be, "device 526 has an RPS problem; does not have a seek problem; does not transmit too much data." This is a series of facts that have been deduced by analyzing large amounts of performance data in a procedural manner. The inference engine will do a very fine job of taking the facts for the entire configuration, such as above, comparing them, and deciding what the facts mean. The example fact above taken on its own is interesting, but probably not sufficient. The analysis of all of the facts in a global context is an area in which the inference engine technology should be very effective. There are other areas of concern, but this is the most significant issue associated with expert system and performance management.

THE MOST SIGNIFICANT ISSUE ASSOCIATED WITH THE INTERFACE OF EXPERT SYSTEMS TO MVS PERFORMANCE MANAGEMENT PRODUCTS IS THE PROPER PREPARATION OF MEASUREMENT DATA IN ORDER TO OPTIMIZE THE EXECUTION OF THE EXPERT SYSTEM.

2a. THE TIME SPECTRUM OF PERFORMANCE MANAGEMENT

Performance management in its entirety consists of the support of
three major areas:

1. REAL TIME
2. CURRENT TIME
3. FUTURE TIME

The REAL TIME environment is the one on the machine room floor. It is analogous to the gauges and controls in your automobile.

The CURRENT TIME environment consists of the tuning of the configuration as it exists on the floor. It does not include the planning for new hardware; it merely involves the optimization of the hardware and software as it exists today. It is analogous to the tuning of your automobile including adding new tires or sparkplugs, and the optimization of the automobile by the garage mechanic. This is more of a batch process and tends to operate against historical, although reasonably current information.

And finally, FUTURE TIME consists of the planning for future equipment acquisitions or changes. This is analogous to planning for your next automobile. How big should it be, what new features are available, what new models are available?

The CURRENT and FUTURE TIME environments can include significant batch reduction of data and creation of reports in a form either on paper or browsed online in some interpretive manner. Also included, to test hypothetical solutions, should be an analytic model to support the "what if" environment. Notice the model is included in the CURRENT TIME situation, since the testing of "what if" resolution to CURRENT TIME tuning issues is important as well. As for FUTURE TIME issues, heretofore we have applied the model to the FUTURE TIME situation, but more and more we are seeing modeling capabilities surface to address CURRENT TIME situations as well.

The application of expert systems to these three environments is certainly feasible. The key, as I discussed previously, is the preparation of data in a form that is acceptable to the expert system. For that reason the most promising initial application of this technology is in the CURRENT TIME and FUTURE TIME environments. Although REAL TIME applications can occur, the time sensitive nature of responding in a real time manner dictates that major improvements in performance monitor technology be created. This is not as big an issue in the CURRENT or FUTURE TIME as batch reduction is an acceptable component of the process.

2b. PERFORMANCE MANAGEMENT EXPERT SYSTEM OPPORTUNITIES

PRODUCTS DEVELOPED WITH EXPERT SYSTEM TECHNOLOGY WILL PROVIDE LABOR-SAVING DEVICES THAT WILL ALLOW SYSTEMS PROGRAMMING EXPERTS TO FOCUS ON REAL ISSUES AND OPPORTUNITIES WORTHY OF THEIR EXPERTISE. EXPERT SYSTEMS TECHNOLOGY WILL NOT REPLACE EXPERTS, BUT WILL ENHANCE THEIR ABILITY TO FOCUS, AND THE NOVICE'S ABILITY TO PERFORM AS AN EXPERT.

In all of the three environments of the time spectrum (REAL TIME, CURRENT TIME, FUTURE TIME), the analysis of the configuration in order to detect contention and pinpoint bottlenecks is the most promising application of this new technology. Heretofore, performance products have produced a lot of information, but require a systems analyst to determine bottlenecks and pinpoint the problems. We believe that expert system technology can be applied to this process. Specific opportunities in each area also include:

1. FUTURE TIME - optimal design and in particular test driving the model in an iterative manner.
2. CURRENT TIME - analyze SRM data and suggest the setup of IPS parameters.
3. REAL TIME - intelligent help facilities for the operator. That is, help facilities that assist in the analysis of detected problems as directed by the operator himself.

It is important to understand that expert system technology will not replace system programmer expertise. What it will do is allow the system to capture some of the knowledge of these experts so that novices or operations personnel can begin to address the operational management of the performance of the data center. This requires that performance data be reduced and put in a form that is understandable to someone who is less highly trained than your average performance analyst. In addition, we see this technology as providing the ability to produce products that will become labor-saving devices for the expert systems programmer. In other words, he will be free to focus on problems that truly require his depth of experience and knowledge rather than the more mundane kinds of things that today cannot be handled by anybody else. For example, the analysis of hundreds of disk drives in order to determine their performance characteristics is something that requires such expertise, but clearly is not performed on a regular basis. The regular execution of an expert system reading DASD performance data can produce focused analysis and advice that the less trained systems personnel can either follow or discuss with the performance expert for his data center.

In the long run, I am confident that expert systems technology can begin to apply process control techniques to the computer center environment although that is not something that will happen in the near future.

3. CONCLUSION

In conclusion then, I believe that expert systems technology is the fifth generation of the evolution of development languages. As such, it provides significant capability for the implementation of new performance management capabilities as well as a tool for addressing the programmer productivity and application backlog issue. It has application across the entire time spectrum of performance management issues, but because of the nature of the input data required, the initial implementations will surface in the CURRENT TIME and FUTURE TIME arenas with REAL TIME implementations requiring improved performance data sources. And finally, that expert systems will not displace experts but will focus them on real issues, and upgrade the level of capability of the novice to perform at an expert level.

EXPANDED STORAGE USAGE AND TUNING

Carolyn Hanna
McDonnell Douglas Aerospace Information Services Company

ABSTRACT

A study was done on the use of expanded storage on a 3090-400. The basis of the study was a problem of high MVS overhead which appeared to result from high use of expanded storage and auxiliary storage. The application in the study in TSO/CADCAM. This paper will present an overview of real, expanded, and auxiliary storage. The tuning parameters for expanded storage and methods for monitoring and measuring the use of expanded storage will be described. The results of tuning and parameter adjustment in an attempt to efficiently use storage and reduce the MVS overhead on the 3090-400 are reported.

INTRODUCTION

The introduction of the 3090 family of processors brought in a new feature, expanded storage. Expanded storage provided the necessary relief for paging and swapping, system performance improved initially. As the load on the system grew, so did the use of expanded storage and migration, and so did the MVS overhead. Overhead is unaccounted CPU time. Migration is the movement of a page from expanded storage to auxiliary storage. The page is moved from expanded storage to central storage and then out to auxiliary storage.

This author began to experiment with the expanded storage parameters in IEAOPTxx in an attempt to tune a heavily loaded system. The tuning environment is described in Figure A. The original tuning goal was to direct more work, mainly batch, to auxiliary storage as a storage crisis was in full swing. By moving pages directly to auxiliary storage the demand on expanded storage and migration would be reduced. This would then reduce any associated MVS overhead. The high use of expanded storage was believed to be causing a high amount of MVS overhead (25% or greater).

After a sharp learning curve which will be shared with the reader, several parameter settings were tested. The results of these adjustments, and an overview of expanded storage are presented in the following sections of this paper.

ENVIRONMENT

```
Processor Model  3090-400
Single Image
Real Storage        64M
Expanded Storage   128M
12 3380 LOCAL Page Datasets
1  3380 PLPA and COMMON
TSO  ave WSS  200K
CADD ave WSS 2000K

MVS2.1.7
JES2.1.5
RMF3.4.1
OMEGAMON v685
EPILOG   v250
```

Figure A

OVERVIEW

Expanded Storage is a hardware feature available on the 3090 family of processors. It is part of the hierarchy of storage which includes central storage, expanded storage, and auxiliary storage.[4] Refer to Figure B. Expanded storage was introduced as an alternative to DASD paging. It is managed by the Real storage Manager (RSM) and the System Resource Manager (SRM).

RSM plays a dual role in the use of expanded storage. One role is to determine which pages go to expanded storage and which go to auxiliary storage. The second role is to manage the migration or removal of pages from expanded storage to auxiliary storage. A page that is migrated from expanded storage to auxiliary storage must first be sent back into central storage before going to auxiliary storage. Page movements involving expanded storage are synchronous and performed a page at a time for both page and swap activities.[9] VIO activity is to auxiliary storage. The following terms will be referenced in the remainder of this paper:

- The Unreferenced Interval Count (UIC) is an indicator of the contention for real storage. The average system-high UIC is a measure of how long a page has remained unreferenced in real storage. The value is in seconds. When the UIC is low, contention for storage is high.[4]

- The Migration Age (MA) is an indicator of the age of a frame in expanded storage. This value is in seconds. It is not an actual count like the UIC, instead it is a calculation done by the SRM based on the length of time required to cycle through the expanded storage table. More accurately, it is an indicator of the demand on expanded storage.

- The Criteria Age (CA) is an estimate of the time a page will remain in central or expanded storage. The Criteria Age is specified in the OPT, and can be set for several types of pages which will be addressed later. This value is used in the selection formula by RSM when deciding to send a page to expanded storage or to auxiliary storage. The larger the criteria age value, the more likely the page is to go to auxiliary storage. The criteria age can be set in the range 0 - 32767. The value is in seconds. The criteria age is a set value, determination of the setting is made by monitoring the migration age and the UIC for the system.

Throughout this paper the terms real, expanded, and auxiliary will be used to refer to real storage, expanded storage, and auxiliary storage respectively.

STORAGE HIERARCHY

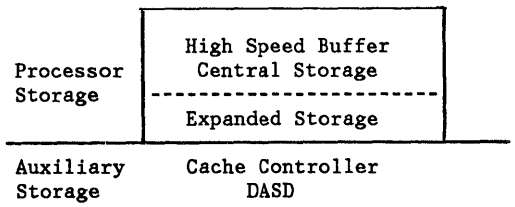


Figure B

OPT KEYWORDS

There are 8 keywords (22 specifications) in the OPT for which the Criteria Age (CA) can be set. Keep in mind the higher the criteria age the more likely a page will be sent to auxiliary storage based on the decision formula.

The CA default values used in the formula are low and always default to expanded storage. These values are displayed in Figure C. From this author's observation of a heavily loaded system, where the page movement rate from central to expanded storage was over 1800 pages a second, the migration age never dropped below 350 seconds. The movement of pages to expanded storage and the migration age are reported in OMEGAMON and RMF. Since the highest default value for criteria age was 100, pages always went to expanded storage.* In such a case the criteria age must be set at 350 or larger to be effective. These parameters must be continually adjusted based on the system load and the migration age being experienced. The low, high, and average migration age for an RMF interval are reported in the RMF Paging Activity Report shown in Figure D.

The OPT keywords are as follows:

ESCTPOC(n) -changed page-out pages
ESCTPOU(n) -unchanged page-out pages
ESCTSTC(n) -changed stolen pages
ESCTSTU(n) -unchanged stolen pages

For the above parameters the decision formula is: MA > CA.
For swap pages the keywords are:
ESCTSWTC(n) -changed swap-out pages
ESCTSWTU(n) -unchanged swap-out pages
ESCTSWWS(n) -working set pages ready for swap-out

For the above parameters the decision formula is: UIC + MA > CA.
An exception to this is ESCTSWWS(2), terminal wait pages are sent to expanded storage when: (UIC + MA) - Think Time > CA.

For the above seven keywords (n) takes on the following meaning:

0 - nonswappable/common/privileged users
1 - all others not 0 or 2
2 - Terminal wait swap users/or TSO users

The eighth keyword is for Virtual Fetch:
ESCTVF - virtual fetch pages
The decision formula for virtual fetch pages is: MA > CA.

*The average UIC=20, Think Time=10 (OMEGAMON)

For clarification, the specification of ESCTPOC(2)=400 in the OPT would set the criteria age at 400 for all changed page-out pages for TSO users. These parameters are outlined in the Initialization and Tuning Guide(5). To simplify the tuning effort, the OPT can be changed dynamically by use of the set command: SET OPT=xx. Setting the OPT during an RMF interval does not cause the interval to reset.

THE DECISION

An indicator of the contention for storage is determined (left side of the inequalities) and compared to the CA. When the inequality is true the page goes to expanded storage, when the two sides are equal or the right side (CA) is greater then the page goes to auxiliary storage.

MONITORING EXPANDED STORAGE

The three tools used to monitor expanded storage usage were OMEGAMON, DEXAN and RMF. These are very similar since OMEGAMON and DEXAN use RMF data.

OMEGAMON allows real time visibility into a dynamic system. The migration age value, migration rate, and various other expanded storage values can be viewed through a series of commands. For monitoring the use of the OPT values and expanded storage this author set up the screen space presented in Figure E. The values displayed are for a 3090-400 before any OPT settings were implemented. This screen space will be revisited in the section on tuning results. The most important point here is that the analyst realize what is being reported. For example the OMEGAMON system paging displays do not include paging to expanded storage. The expanded storage displays must be viewed separately. Therefore, in Figure E, the SPAL does not include paging to expanded storage. The page fault rate and page delay time (RPAG,RPDL) are only reporting auxiliary storage values.

When the CA values are in effect, viewing the MPAGxx and MPGIxx displays will reveal the effects on the paging packs. The MPAGxx reports only auxiliary storage paging. The SRCT display is valuable, specifically SRCT.R, to see the rate at which pages are sent to auxiliary, expanded, or migrated from expanded for various swap reasons.[3] The 'ESMG' command in Figure E displays the current OPT settings. When a setting is invoked, an asterisk is displayed next to it. When the MA is below the set value, it will not be invoked until a decision is made involving the specified type of page. For example, you may set CA for batch, if there is no batch on the system it is not invoked. If you have OMEGAMON V685 the ESMG display may be incorrect (BATCH and TSO are reversed) and a zap is available to fix the problem. It is important to check this before you set CA values since it could be confusing as to which values are actually in effect. The impact of sending all swap for batch to auxiliary storage is much different than sending all swap for TSO. For individual users, the ESFC - frame count value will display the number of expanded storage frames a user has. A swap to expanded storage is not included in the swap count however.
DEXAN reports the wait reasons and time for a performance group or period. EPILOG gives this visibility also. The DEXAN interval can be adjusted within the RMF interval. This tool is helpful to determine the effect of paging and swapping on a given performance group.

RMF Monitor 1 provides two reports which include expanded storage activity. The RMF Paging activity report, Figure D, reports key indicators of expanded storage usage on the right side of the report. The paging values on the left side of the report are for auxiliary storage. The Swap Placement Activity report is also helpful in tracking the effects of changes, particularly if you are attempting to send pages to auxiliary storage. RMF Monitor II - The Address Space State Data (ASD) report includes an Expanded Storage

TABLE of OPT Parameters - Default values

		SWP-UN	SWP-CH	SWP-WKST	STOLE-UN	STOLE-CH	PGOUT-UN	PGOUT-CH
0	NSW/CMN	100	100	100			100	100
1	BATCH	100	100	100	20	20	100	100
2	TSO	60	60	50	15	15	100	100
	VIRTUAL FETCH							

Omegamon Display
Figure C

P A G I N G A C T I V I T Y PAGE 1

OS/VS2 SP2.1.7 | SYSTEM ID U1LU | RPT VERSION 3.4.1 | DATE 03/26/87 | TIME 09.59.02 | INTERVAL 59.58.523 | CYCLE 1.000 SECONDS

OPT = IEAOPTE1

MAIN STORAGE PAGING RATES PER SECOND

CATEGORY	PAGE RECLAIMS NON SWAP	% OF TOTL NON SWAP	PAGE IN SWAP	PAGE IN NON SWAP	PAGE IN TOTAL RATE	% OF TOTL SUM	PAGE OUT SWAP	PAGE OUT NON SWAP	PAGE OUT TOTAL RATE	% OF TOTL SUM
PAGEABLE SYSTEM AREAS (NON VIC.										
LPA	0.10	1		0.12	0.12	0				
CSA	0.13	1		0.11	0.11	0		0.00	0.00	0
SUM	0.23	1		0.23	0.23	0		0.00	0.00	0
ADDRESS SPACES										
VIO	0.26	1		1.03	1.03	1		2.18	2.18	7
NON VIO	19.27	98	42.93	83.96	126.89	99	27.13	0.21	27.34	93
SUM	19.53	99	42.93	84.99	127.92	100	27.13	2.39	29.52	100
TOTAL SYSTEM										
VIO	0.26	1		1.03	1.03	1		2.18	2.18	7
NON VIO	19.49	99	42.93	84.19	127.12	99	27.13	0.21	27.34	93
SUM	19.7c	100	42.93	85.22	128.14	100	27.13	2.39	29.52	100

REAL STORAGE MOVEMENT TOTAL RATE 204.90

RATE OF PAGE MOVEMENT TO ES 1,654.23

MIGRATION RATE 101.70

AVAILABLE ESF

MIN	0
MAX	1113
AVG	202

	HIGH UIC	MIGR AGE
MIN	3	312
MAX	17	445
AVG	9.0	355.8

ESF CONFIGURATION

INSTALLED	ONLINE
65536	65536

MAIN STORAGE FRAME COUNTS

	MIN	MAX	AVG
TOTAL FRAMES			
LPA	412	653	554
CSA	686	879	787
PRIVATE AREA	22,354	23,936	23,313
UNUSED	195	840	300
TOTAL	31,890	31,890	31,890
FIXED FRAMES			
SQA	1,505	1,520	1,513
LPA	55	57	55
CSA	181	190	183
LSQA	4,887	5,985	5,420
PRIVATE AREA	680	1,231	795
BELOW 16 MEG	916	1,469	1,027
TOTAL FIXED	7,375	8,560	7,968

NUCLEUS FRAMES AT END OF INTERVAL 689

TOTAL ONLINE SYSTEM FRAMES AT END OF INTERVAL 32,579

NUMBER OF SAMPLES 360

LOCAL PAGE DATA SET SLOT COUNTS

	MIN	MAX	AVG
AVAILABLE SLOTS	301,987	315,286	306,449
VIO SLOTS	24	4,285	3,089
NON-VIO SLOTS	170,690	180,057	176,460
BAD SLOTS	0	0	0
TOTAL SLOTS	486,000	486,000	486,000

Figure D

Frame (ESF) value. The Paging report (SPAG) reports migration rate and age values, expanded storage movement rate, and expanded storage available.

THE TUNING EFFORT

The system activity, before any keyword settings, is presented in Figure E. Note the rate of page movement to expanded storage and the migration age. Note also the paging activity reported. Hundreds of parameter combinations could be tested, fifteen OPT settings were implemented and observed, seven were selected for discussion. Data on these different settings is reported in Appendix A. Each setting is referred to as a case with Case 0 being the initial condition of the system before any OPT changes were implemented. The data for Case 0 is in Table 0A of Appendix A and for Case 1 in Table 1A and so on. The data presented in Appendix A was averaged for the intervals for which the keywords were in effect. A number of observations for each setting were recorded during peak periods. A general discussion will be presented here for each case.

Three OPT settings were implemented for n=1, or 'all other' category and are presented in Case 1 - Case 3. Case 7 includes both the 'all other' and TSO and is discussed with the TSO settings. The terms BATCH and 'all other' both refer to n=1.
In Case 1, the keywords in effect were for changed and unchanged page-outs. This setting had no apparent impact to the system in general. The number of batch initiators was reduced to six and therefore the batch load on the system was light. It should be noted that when (n) is 1, batch (ie. PGN 1) may not be the only jobs which are affected. Determination of what address spaces are classified 'all other' is important to avoid unexpected degradation. The CPU utilization was low in Case 1 and batch throughput was fairly good. Minimal paging was observed for the batch jobs. The overhead decreased from 26% (Case 0) to 17%. Note that the migration rate remained high and the page move rate to expanded was reduced over Case 0.

In Case 2 the keywords in effect were for changed and unchanged swap pages and for working set pages. The impact to the system was minimal. The overhead decreased from 26% (Case 0) to 20%. The migration rate was less as was the page movement rate to expanded storage. CADD first period response time increased by .1s.

In Case 3 both swapping and paging keywords were in effect. The overhead increased from 26% (Case 0) to 27%. The response time for the paging packs doubled. CADD first period response time increased by .2s.

Three OPT settings were implemented for n=2, or TSO, and are presented in Case 4 - Case 6.
In Case 4 the page-out keywords were in effect. Due to the direct use of auxiliary storage there was considerable less expanded storage page movement. The migration rate was less due to the reduced demand for expanded storage, and the MVS overhead decreased from 26% (Case 0) to 19%. CADD first period response time was increased .3s.

In Case 5 swap keywords were in effect. The impact of sending swap trim and swap working set pages to auxiliary storage was quick and severe. As shown in Table 5A the response time is twice its normal value. A large amount of the reported value is spent in page-in waits. The MVS overhead increased from 26% (Case 0) to 30%. The migration rate for this interval was zero. Since pages were being sent directly to auxiliary, the demand for expanded storage was decreased, therefore, there was no need to migrate pages to free up expanded storage to meet the demand.

In Case 6 both the page-out keywords and the swap keywords were implemented. As in Case 5, the impact of this OPT setting was immediate and severe. First, all paging and swapping activity went to auxiliary storage. Since no swap datasets exist the page requests were delayed by the swap activity. The number of local paging devices had been reduced when we brought in expanded storage, so the current configuration was not able to handle the increased load. These delays were noticeable to all users. CADD period 3 was most noticeable since they were swapped in this period. Although this setting was only on the system for a few minutes, calls came in from users about response time. According to DEXAN, 30% of the response time was due to page-in waits. (the page-in waits reported are only auxiliary page-ins) The MPAG command revealed the CADD users paging rate as high as 524 pages/sec. The paging device activity is reported in Appendix A. This activity appears to be due to the swapping which is going to auxiliary storage. Since the swap trim is greater, more paging is experienced. Although users do not experience swap out delay, the lack of swap data sets was again a factor for the page-outs/ins, which the user did experience. The MVS overhead increased from 26% (Case 0) to 29%.

In Case 7 all BATCH pages and page-outs for TSO went to auxiliary. This setting had the effect of disengaging expanded storage for BATCH. The expanded storage move rate was considerably reduced, the MVS overhead decreased from 26% (Case 0) to 25%. The response time for BATCH and TSO increased significantly. Period three response time was extremely high as users are effected by the demand on auxiliary storage and the batch swap-outs to the paging devices. In Case 7 the migration age value was observed through OMEGAMON to change at the approximate rate of 1.5sec. per sec. This generally occurred when the migration rate was zero, however, it was also observed when migration was taking place. This same rate of change in the migration age was observed in Cases 3, 5, and 6. In the other Cases the migration age would remain constant for several moments, change, then remain constant for several more minutes.

CONCLUSION

Many OPT settings were tested, most of which would not be implemented since the purpose of expanded storage is to reduce delays caused by paging and swapping. The purpose of the settings was to gain a better understanding of how expanded storage works. Two of the settings, Case 1 and Case 2 were used in the CADD environment, with Case 1 providing the best trade-off in CPU usage and performance of the online system. In general, the amount of CPU cycles gained due to reduced MVS overhead is not worth the cost of online degradation. This author found that Migration Age is a somewhat erroneous value. It seems odd to set a Criteria Age based on a value that is not an actual count or time but a calculated value (by SRM). More documentaion is needed on how migration age is determined and adjusted. The current keywords in the OPT are far too broad to offer much control, sort of an all or none choice. A preferred option would be to select the type of user, based on performance group, jobname or address space, to use auxiliary storage and not expanded.

```
                                                                    UNIT=504
__________ CSHEXP   DED PRT OM/DEX * V685.65 U1LU 3/26/87 10:43:05
SYS     >> IPS=CT,OPT=E1,ICS=CT SYSRES=(MVSU01,DC8) <<
FAFQ        254     1016 K
RCT      _____ FIELD _______              __VALUE__ ___LOW___ __HIGH___
RUIC    UNREFERENCED INTVL CNT                    3         2         4
RASQ    ASM QUEUE LENGTH                         43      1000      1000
RPAG    PAGE FAULT RATE                          98      1000      1000
RDPG    DEMAND PAGING RATE                       96       649       819
RCPU    CPU UTILIZATION                          98        99       100
RCPD    CPU UTIL (DEMAND PAGING)                 98        95        98
RPDL    PAGE DELAY TIME                         112      1000      1000
RPDD    PAGE DELAY (DEMAND PAGING)              112       100       130
SMPP    SWAP PAGE DELAY TIME (MS)
THNK    SYSTEM THINK TIME                  10 SEC    10 SEC    10 SEC

>ESPO   TOTAL NUMBER OF PAGES SENT TO EXPANDED STORAGE
 ESPO.R  2062.8
>ESMV   TOTAL NUMBER OF PAGES MIGRATED TO AUXILIARY STORAGE
 ESMV.R   118.1
 ESMG  MIGRATION AGE =  328 SEC.   MIGRATION RATE = 118.1 PAGES/SEC
+      SELECTION CRITERIA
+        SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
+ NSW/CMN   100      100      100                        100       100
+ TSO       100      100      100       20        20      100       100
+ BATCH      60       60       50       15        15      100       100
+ VIRTUAL FETCH

>THESE DO NOT INCLUDE EXPANDED STORAGE

>SCAL   SWAP COUNTS: TOTAL PHYSICAL SWAPS
 SCAL.R     6.2
>SPWS   PAGING: DUE TO SWAPPING
 SPWS.R    46.8
>SPAL   PAGING: TOTAL SYSTEM  (SPAL = SPS + SPSV + SPWS)
 SPAL.R   132.8
>SPS    PAGING: SYSTEM AREA (PRIVATE AND COMMON)
 SPS .R    85.9
>SPSV   PAGING: VIO
 SPSV.R
>SRCT   DISPLAY SWAP REASON COUNTS BY TYPE
 SRCT.R   TOTAL   TOTAUX  PSTOAUX  LSTOAUX    TOTES   PSTOES   LSTOES  ESTOAUX
+   TI       .8                                  .8                .8
+   LW      1.2                                 1.2       .4        .8
+   DW       .4                                  .4                .4
+TOTAL      2.5                                 2.5       .4      2.1


>MPAG05 DISPLAY MULTIPLOT OF PAGING RATES GREATER THAN NN PAGES/SECOND
 MPAG05|__TASK__ PG/S 0_2_4_6_8|PERF GRP PG/S 0_2_4_6_8|VOLUME PG/S 0_2_4_6_8|
+      |TSOGTGS  24.4 ->. . . .|PS300    59.9 --->. . .|MVPE30   .0 > . . . .|
+      |TSOGMAL   5.9 > . . . .|                       |MVPE30   .3 > . . . .|
+      |TOT PAG: 58.9 --->. . .|                       |MVPE31 19.4 ->. . . .|
+      |                       |                       |MVPE32 27.4 --> . . .|
+      |                       |                       |MVPE33 20.2 ->. . . .|
+      |                       |                       |MVPE35 18.9 ->. . . .|
+      |                       |                       |MVPE38 17.7 ->. . . .|
+      |                       |                       |MVPE39  6.7 > . . . .|
+      |                       |                       |MVPE3A  5.9 > . . . .|
+      |                       |                       |MVPE3B  8.8 > . . . .|
+      |                       |                       |MVPDC0 10.5 ->. . . .|
+      |                       |                       |MVPDC2 11.3 ->. . . .|
+      |                       |                       |MVPDC4  3.3 > . . . .|
+      |                       |                       |MVPDC6 16.4 ->. . . .|
+      |                       |                       |TOTAL:166.9 ---===>>>|
```

O M E G A M O N
! CANDLE CORPORATION

Figure E

SUMMARY

A brief description of expanded storage and some key terms and concepts have been presented. Some of the tools available which offer visibility into expanded storage use have been discussed. A tuning effort using the OPT expanded storage keywords was presented. The conclusion of this author is that little control exists for expanded storage and setting the available parameters in the OPT for the CADD environment were effective in some ways and degrading in others. Better controls of expanded storage are required for its future cost effective use.

REFERENCES

1] Anderson, J. W., King, G. M., and Singh, Y., IBM 3090 Performance: A Balanced System Approach. IBM Systems Journal v25, no1, 1986.

2] Beretvas, Thomas, Analysis of Processor Storage and Paging Configurations, CMG transactions, 1986,p 772/788.

3] Candle Corp., Omegamon/MVS MVS/XA Reference Manual, V685, 198x.

4] IBM Corp., OS/VS Performance Notebook, GC28-0886-2, 1983.

5] IBM Corp., MVS/XS Initializaiton and Tuning Guide, GC28-1149-3, June 1986.

6] IBM Corp., IBM 3090 Model 200 Processor Complex Installation Notebook, GG66-0213-00, July 1985.

7] IBM Corp., MVS/XA RMF Reference and User's Guide, LC28-1138-2, July 85.

8] Lee, Micheal, Candle Performance Conferance Seminar on Expanded Storage, 5/14/87.

9] Tucker, S. G., IBM 3090 System: An overview, IBM Systems Journal, v25 no.1, 1986.

CASE# 0

TABLE 0A

```
 MIGRATION AGE = 355  SEC.   MIGRATION RATE = 101.7 PAGES/SEC
 SELECTION CRITERIA
         SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
0)NSW/CMN   100      100      100                        100       100
1)BATCH     100      100      100       20        20      100       100
2)TSO        60       60       50       15        15      100       100
  VIRTUAL FETCH

   EXP MOVE RTE       1654.2    CPU UTIL(3090-400)     98%
   PAGE DELAY(DEMAND)   80      OVERHEAD               26%
   SYSTEM PAGING RTE    72.8    SYSTEM/MASTER ADDR      9%
   PAGING DUE TO SWAP   67.2    TSO/CADD 258-page/sec 23.6 resp   .5s
   UIC                   8      BATCH     12-page/sec 12.5 resp  5.1m
   TNK TIME             10      PAGE DEVICES#13 Util  19.5 Resp 33ms
   PAGE FAULT RATE     107
   OBSERVATIONS:
   Period 1 TSO/CADD Private page-in-wait = .04s
                     Swap    page-in-wait = .01s
   Period 3 TSO/CADD resp. = 8.1s
```

CASE# 1

TABLE 1A

```
 MIGRATION AGE = 720  SEC.   MIGRATION RATE = 97.1 PAGES/SEC
 SELECTION CRITERIA
         SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
0)NSW/CMN   100      100      100                        100       100
1)BATCH     600      600      600      400       400     750*      750*
2)TSO        60       60       50       15        15      100       100
  VIRTUAL FETCH

   EXP MOVE RTE       1224.0    CPU UTIL(3090-400)     87%
   PAGE DELAY(DEMAND)   80      OVERHEAD               17%
   SYSTEM PAGING RTE    72.8    SYSTEM/MASTER ADDR      5%
   PAGING DUE TO SWAP   34.2    TSO/CADD 242-page/sec 35.4 resp  .5s
   UIC                  17      BATCH      5-page/sec 12.5 resp 1.46m
   TNK TIME             10      PAGE DEVICES#13  Util 20.6 resp 41ms
   PAGE FAULT RATE      91
   OBSERVATIONS:
   Period 1 TSO/CADD Private page-in-wait = .11s
                     Swap    page-in-wait = .01s
   Period 3 TSO/CADD resp. = 7.7s
```

CASE# 2

TABLE 2A

```
 MIGRATION AGE = 634  SEC.   MIGRATION RATE = 58.9 PAGES/SEC
 SELECTION CRITERIA
         SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
0)NSW/CMN   100      100      100                        100       100
1)BATCH     700*     700*     700*      20        20      100       100
2)TSO        60       60       50       15        15      100       100
  VIRTUAL FETCH

   EXP MOVE RTE        852.6    CPU UTIL(3090-400)     89%
   PAGE DELAY(DEMAND)   70      OVERHEAD               20%
   SYSTEM PAGING RTE   101.6    SYSTEM/MASTER ADDR      7%
   PAGING DUE TO SWAP   72.2    TSO/CADD 239-page/sec 11.2 resp  .6s
   UIC                  52      BATCH     10-page/sec  3.5 resp 3.1m
   TNK TIME             10      PAGE DEVICES#13  Util 15.6 resp 35ms
   PAGE FAULT RATE      32
   OBSERVATIONS:
   Period 1 TSO/CADD Private page-in-wait = .01s
                     Swap    page-in-wait = .01s
   Period 3 TSO/CADD resp =  8.7s
```

CASE# 3

TABLE 3A

```
 MIGRATION AGE = 454  SEC.   MIGRATION RATE =  90.9 PAGES/SEC
 SELECTION CRITERIA
         SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
0)NSW/CMN   100      100      100                       100       100
1)BATCH     500*     500*     500*      300       300     500*      500*
2)TSO        60       60       50       15        15      100       100
  VIRTUAL FETCH

   EXP MOVE RTE       1394.7    CPU UTIL(3090-400)     99%
   PAGE DELAY(DEMAND)  123      OVERHEAD               27%
   SYSTEM PAGING RTE   291.8    SYSTEM/MASTER ADDR     10%
   PAGING DUE TO SWAP  174.2    TSO/CADD 247-page/sec 65.3 resp  .7s
   UIC                  26      BATCH     11-page/sec 12.5 resp 2.15m
   TNK TIME             10      PAGE DEVICES#13 Util  36.2 Resp 66ms
   PAGE FAULT RATE      74
   OBSERVATIONS:
   Period 1 TSO/CADD Private page-in-wait = .05s
                     Swap    page-in-wait = .02s
   Period 3 TSO/CADD resp. = 9.2s
```

CASE# 4

TABLE 4A

```
 MIGRATION AGE = 492  SEC.   MIGRATION RATE = 29.3 PAGES/SEC
 SELECTION CRITERIA
         SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
0)NSW/CMN   100      100      100                       100       100
1)BATCH     100      100      100       20        20      100       100
2)TSO        60       60       60       15        15      600*      600*
  VIRTUAL FETCH

   EXP MOVE RTE        621.3    CPU UTIL(3090-400)     89%
   PAGE DELAY(DEMAND)  113      OVERHEAD               19%
   SYSTEM PAGING RTE   102      SYSTEM/MASTER ADDR      4%
   PAGING DUE TO SWAP   59      TSO/CADD 239-page/sec  23.5resp  .9s
   UIC                  46      BATCH      6-page/sec   1.7resp 3.6m
   TNK TIME             10      PAGE DEVICES#13  Util 48.4 resp 56ms
   PAGE FAULT RATE      37
   OBSERVATIONS:
   Period 1 TSO/CADD Private page-in-wait = .01s
                     Swap    page-in-wait = .01s
   Period 3 TSO/CADD resp =  9.8s
```

CASE# 5

TABLE 5A

```
 MIGRATION AGE = 499  SEC.   MIGRATION RATE =  0.0 PAGES/SEC
 SELECTION CRITERIA
         SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
0)NSW/CMN   100      100      100                       100       100
1)BATCH     100      100      100       20        20      100       100
2)TSO       600*     600*     600*      15        15      100       100
  VIRTUAL FETCH

   EXP MOVE RTE         60.6    CPU UTIL(3090-400)     78%
   PAGE DELAY(DEMAND)  452      OVERHEAD               30%
   SYSTEM PAGING RTE  1032      SYSTEM/MASTER ADDR      6%
   PAGING DUE TO SWAP  744      TSO/CADD 239-page/sec 277.2resp 1.2s
   UIC                  48      BATCH      7-page/sec  3.5 resp 3.5m
   TNK TIME             10      PAGE DEVICES#13  Util 24.3 resp 163ms
   PAGE FAULT RATE     152
   OBSERVATIONS:
   Period 1 TSO/CADD Private page-in-wait = .19s
                     Swap    page-in-wait = .02s
   Period 3 TSO/CADD resp. = 11.5s
```

CASE# 6

TABLE 6A

```
 MIGRATION AGE = 498  SEC.   MIGRATION RATE =   0.0 PAGES/SEC
 SELECTION CRITERIA
         SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
0)NSW/CMN   100      100      100                        100       100
1)BATCH     100      100      100       300       300      100       100
2)TSO       650*     650*     650*      450       450      700*      700*
  VIRTUAL FETCH

   EXP MOVE RTE        298.5    CPU UTIL(3090-400)     68%
   PAGE DELAY(DEMAND) 1080      OVERHEAD               29%
   SYSTEM PAGING RTE  1055      SYSTEM/MASTER ADDR      6%
   PAGING DUE TO SWAP  816      TSO/CADD 246-page/sec 199. resp 2.4s
   UIC                  40      BATCH      5-page/sec   7.5resp 4.2m
   TNK TIME             10      PAGE DEVICES#13  Util 36.8 resp 82ms
   PAGE FAULT RATE     174
   OBSERVATIONS:
   Period 1 TSO/CADD Private page-in-wait = .74s
                     Swap    page-in-wait = .11s
   Period 3 TSO/CADD resp =  15.8s
```

CASE# 7

TABLE 7A

```
 MIGRATION AGE = 397  SEC.   MIGRATION RATE =   24.3 PAGES/SEC
 SELECTION CRITERIA
         SWP-UN   SWP-CH SWP-WKST STOLE-UN STOLE-CH PGOUT-UN PGOUT-CH
0)NSW/CMN   100      100      100                        100       100
1)BATCH     650*     650*     650*      450*      450*     700*      700*
2)TSO        60       60       50        15        15      400*      400*
  VIRTUAL FETCH

   EXP MOVE RTE        858.7    CPU UTIL(3090-400)     74%
   PAGE DELAY(DEMAND)  293      OVERHEAD               25%
   SYSTEM PAGING RTE   853      SYSTEM/MASTER ADDR      9%
   PAGING DUE TO SWAP  703      TSO/CADD 263-page/sec 132.5 resp .7s
   UIC                  20      BATCH      6-page/sec  17.6 resp 7.4m
   TNK TIME             10      PAGE DEVICES#13  Util  55.5 resp 120ms
   PAGE FAULT RATE     155
   OBSERVATIONS:
   Period 1 TSO/CADD Private page-in-wait = .15s
                     Swap    page-in-wait = .01s
   Period 3 TSO/CADD resp = 25.8s
```

MEASUREMENT AND ANALYSIS OF EXPANDED STORAGE IN IBM 3090-400 SYSTEMS

Jeffrey P. Buzen
J. William Mullen
Annie W. Shum

Abstract

This paper presents a set of measurements which characterize expanded storage performance under varying loads. The measurements, which are based on a 3090-400 system operating in a heavy TSO environment, illustrate the nature of page migration and its effect on overall performance. The role of the Expanded Storage Criteria Table in controlling access to expanded storage and migration to auxiliary storage is also discussed.

OVERVIEW OF EXPANDED STORAGE

Expanded storage is one of the most important new features of the IBM 3090 architecture. Essentially, expanded storage can be regarded as an ultra high speed paging subsystem. When the real storage manager (RSM) decides to move a page out of central storage, the page can go either to a conventional auxiliary storage device or to expanded storage. Similarly, when a page must be brought into central storage, it can come from auxiliary storage or from expanded storage.

As this description indicates, expanded storage can only be used for 4K block transfers to and from central storage. Individual bytes in expanded storage cannot be addressed directly, and direct I/O transfers between expanded storage and conventional auxiliary storage cannot occur. Figure 1 illustrates the way expanded storage is integrated into an IBM 3090-400 system.

Since access to expanded storage is controlled by the RSM, expanded storage is not directly visible to user programs. Thus, expanded storage may affect the execution time of a user program, but in all other respects expanded storage is completely transparent to the user.

One very important aspect of expanded storage is its synchronous mode of operation. Apparently, two special instructions have been added to the 3090 instruction set to facilitate movement of page frames to and from central storage. The RSM executes these instructions to cause transfers to occur. In contrast with conventional I/O operations to auxiliary storage devices, the CPU is busy during the entire transfer, and no attempt is made to switch the CPU to another instruction stream while the transfer is being carried out.

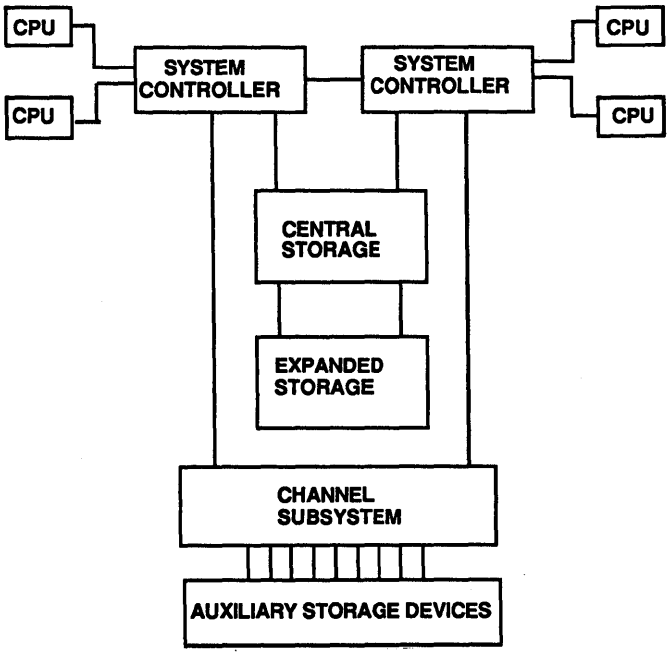


Figure 1

The time to resolve a page fault for a page located in expanded storage has been reported to be approximately 75-80 microseconds [1]. This compares with an expected time of 2-20 milliseconds to resolve a page fault from auxiliary storage. In a swapping operation where many pages are being transferred at once, the time per page is reported to remain at 75-80 microseconds. In either case, expanded storage transfers are more than an order magnitude faster than transfers to auxiliary storage. There is also a saving in processor overhead for I/O initiation and subsequent handling of the I/O completion interrupt.

One important reason for the development of expanded storage is the need to extend the addressability range of the MVS/XA architecture. Conventional storage with 31-bit addressing is limited to 2 gigabytes (2K megabytes) of addressability. Expanded storage is page addressable with 32-bit addressing. This allows for a 16K gigabyte addressing range, which is 8K times the addressability of central storage.

Other factors have also influenced the decision to develop expanded storage. One consideration is that expanded storage is less costly to fabricate since: addressing is at the 4K block level; there is no direct I/O to conventional auxiliary storage; there is no memory protection hardware. Also, error detection and correction has been enhanced by increasing the number of bits used for this purpose. This factor becomes more important as storage size increases.

EXPANDED STORAGE IMPLEMENTATION

Each new feature implemented in hardware or software brings with it the potential need for changes in resource management algorithms: usually new rules and new parameters are provided to control the use and operation of each new facility. Expanded storage is no exception.

One important change that expanded storage has initiated in MVS is the introduction of a page migration mechanism. To illustrate this mechanism, suppose the RSM decides to remove a page from central storage at a time when there is no space available in expanded storage. In the past, MVS would have sent that page to the next fastest auxiliary storage device. This alternative is still available. Now, however, MVS also has the option of removing old pages from expanded storage and writing them out to auxiliary storage to make room for new outbound pages. This movement of old pages is known as page migration.

Page migration represents additional overhead since pages that were originally sent to expanded storage have to be transferred back into central storage before they can be written out to an auxiliary storage device. In effect, migration adds approximately 150-160 microseconds of CPU overhead per page (i.e., one round trip between central and expanded storage). However, page migration can also provide a substantial benefit by maintaining in expanded storage those pages that have been least recently used and thus are most likely to be referenced again.

As already noted, two choices are possible if pages are being removed from central storage at a time when expanded storage is nearly full: either send the outbound pages directly to auxiliary storage, or send them to expanded storage and initiate a migration. MVS allows system programmers to influence this choice and provide certain types of pages preferential treatment in their access to expanded storage.

The choice is influenced through the IEAOPTxx parameters in the Expanded Storage Criteria Table. To describe the role of this table, it is necessary to review the definitions of two important indicators of storage utilization: the unreferenced interval count (UIC), which measures contention for central storage, and the migration age (MA), which measures contention for expanded storage.

UNREFERENCED INTERVAL COUNT (UIC)

The primary indicator of contention for real storage in MVS is the page unreferenced interval count (UIC). A separate UIC value is maintained by the System Resources Manager (SRM) for each occupied real storage page frame. Updating of the UIC values is based upon reference and usage of the frame. At initial allocation of a real storage page frame, the UIC value for that frame is set to zero seconds. As the SRM reviews usage of real storage frames, the UIC value for a given frame is increased by the number of seconds that have elapsed since the frame was last referenced.

To enable the UIC values to drive various MVS memory management algorithms, two additional values are calculated. The system high UIC is the largest of the UIC values in the system at any given instant. For an interval of time, the corresponding measure of contention is the average system high UIC. If these values are large, then contention for real storage is low since, on the average, page frames may remain allocated and unreferenced for long periods of time without being stolen or swapped, and without creating a real storage frame shortage. Similarly, a low value of the system high or average system high UIC indicates that contention for real storage is high.

MIGRATION AGE (MA)

The availability of expanded storage in 3090 processors created a requirement for additional measurements to determine expanded storage utilization and to provide a basis for access to this resource. The primary indicator used to support these functions is the Migration Age (MA). The only information currently provided by IBM on the calculation of migration age is the general statement that migration age represents the average page life of a page residing on expanded storage.

We can derive inferences regarding migration age from the measurement data available in RMF and other sources. First, migration age appears to be a global value rather than set of individual values computed separately for each expanded storage page frame. This assumption is based upon the fact that, at the first occurrence of migration, the migration age is reduced dramatically. The calculation of migration age may be based upon the amount of time that elapses between occurrences of migration and the average number of frames occupied in expanded storage between these occurences. This is analogous to calculations made by VM/370 memory management algorithms.

The migration age value is not limited to the 255 maximum imposed upon the UIC values. Migration age values in the hundreds of thousands have been observed in MVS/XA systems, provided these systems have been operational for extended periods of time and there has been no migration of expanded storage pages to auxiliary storage.

RMF provides measurement interval averages for migration age and also provides the ability to trace the migration age value via the MCVMGAGE trace keyword. Analysis of RMF traces during periods when no migration was occurring has indicated that migration age increases at the rate of 1.5 seconds for each wall clock second of elapsed time. This method of increasing migration age would account for the sizable migration age values observed in systems that processsed for extended periods of time without experiencing migration.

USE OF UIC AND MIGRATION AGE

The rules used by MVS to manage expanded storage usage and page migration allow different categories of pages to be treated differently. One of the factors used as a basis for different treatment is class of storage. The classes are defined as follows:

Class	Description
0	Non-swappable, Common and Privileged
1	All Others Not 0 or 2
2	TSO Swap and TSO other

These classes of storage are used as column headings in a new table, the expanded storage Criteria Table (ESCT), as shown in Figure 3. Algorithms operating on this table determine eligibility for transfer of a task's pages to expanded storage. This table is analogous to the Logical Swap Control Table (LSCT) implemented with the logical swapping facility in MVS.

For non-swap pages (rows 1 and 2), stolen pages (rows 3 and 4) and IMS virtual fetch pages (last row), the current value of Migration Age must be greater than the Criteria Age (MA > CA) that appears in the appropriate row and column of the ESCT for the page to be eligible for transfer to expanded storage. The objective of this basic rule is to prevent expanded storage migration if there is contention for expanded storage frames (as indicated by a low migration age).

For swap pages that are not terminal wait pages (rows 5-7), the UIC plus the Migration Age be must be greater than the appropriate ESCT Criteria Age (UIC + MA > CA) for the page to be eligible for transfer to expanded storage. In this case, increased contention for real storage (indicated by low UIC) can cause pages to be sent directly to auxiliary storage even though contention for expanded storage is only moderate. In other words, non-swap pages can be given preferential treatment over swap pages when contention for real storage increases grows.

For swap pages that are terminal wait pages (row 8), the UIC plus the Migration Age minus the user's 'think time' must be greater than the appropriate ESCT Criteria Age (UIC + MA - TT > CA) for the page to be eligible for transfer to expanded storage. Since think time is subtracted from UIC + MA, this rule gives preferential treatment to TSO users who have shorter think times.

The Criteria Ages are specified in the IEAOPTxx member of SYS1.PARMLIB. Each of the parameters are prefixed with ESCT and have three values corresponding to classes of storage 0, 1 and 2. The parameters and their abbreviations are listed below in Figure 2.

Paged-out Changed	(POC)
Paged-out Unchanged	(POU)
Stolen Changed	(STC)
Stolen Unchanged	(STU)
Swap-out Changed	(SWC)
Swap-out Unchanged	(SWU)
Swap Working Set	(SWWS)
Virtual Fetch (IMS)	(VF)

PAGE TYPES AND THEIR ABBREVIATIONS

Figure 2

Each of these eight page types is assigned three criteria ages, corresponding to the three classes of storage. These criteria ages are then used in determining eligibility for use of expanded storage. The structure of the ESCT and the default criteria ages (in seconds) are shown in Figure 3.

From the default values, it can be seen that MVS/XA stolen pages for common and privileged users are the most favored pages for use of expanded storage. The next favored class includes virtual fetch pages as well as stolen working set pages for TSO users.

Very little has been published to provide guidelines or recommendations on possible changes to the OPT default values in the ESCT. However, early experience indicates that very large changes in these values are required if there is to be any real impact on performance.

CRITERIA TABLE

	Storage Class Priv 0	1	TSO 2	Criteria for Transfer to Expanded Storage
PAGED-OUT				
Changed	100	100	100	MA > CA
Unchanged	100	100	100	MA > CA
STOLEN				
Changed	0	20	15	MA > CA
Unchanged	0	20	15	MA > CA
SWAP-OUT				
Changed	100	100	60	UIC+MA > CA
Unchanged	100	100	60	UIC+MA > CA
WORKING SET				
Ready	100	100	50	UIC+MA > CA
Terminal Wait				UIC+MA-TT > CA
VIRTUAL FETCH				
IMS Only		15		UIC+MA > CA

Figure 3

CASE STUDY

We have discussed the basics of the IBM 3090 architecture and the implementation of expanded storage. We will now examine some actual measurement data and look specifically for indications of the 'Large Systems Effect'.

Our graphs are derived from measurement data obtained from a 3090-400 processor complex operating in single image mode. This complex has 128 megabytes of central storage and 256 megabytes of expanded storage. The primary workload, which is TSO, averages 520-560 logged-on users with approximately 380 of these users active at any given time. There is minimal batch processing allowed on the system. TSO is a production workload, and users tend to be slightly higher in their resource consumption per transaction than users in a TSO development environment would be.

The first chart in this series, Figure 4, illustrates the expected depletion of central storage page frames as the number of logged-on TSO users increases. The points on this and other charts are average values from several measurement intervals. A point of significance in the chart is the values for available central storage frames in the range of 150-170 logged-on users. The slight increase in available frames could be an indication that MVS/XA is beginning to page steal to replenish the available frame queue. An average value of less than 1000 frames is an indication that, during a majority of the measurement intervals, the available frame count was not sufficient to satisfy the demand for central storage.

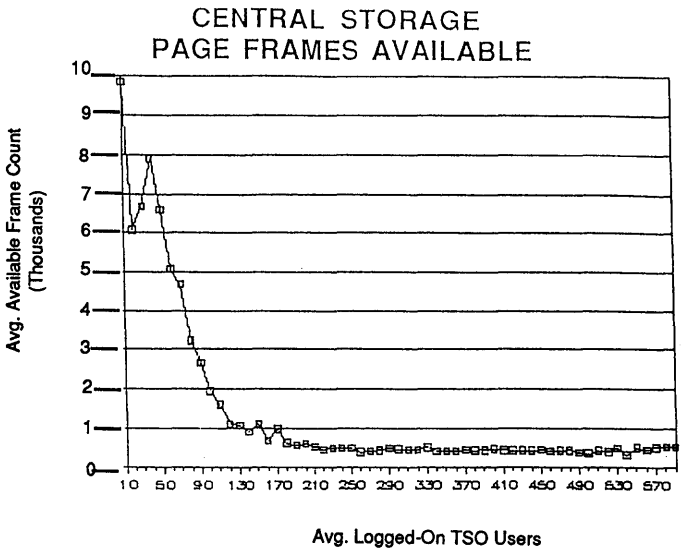


Figure 4

The transfer of pages from central storage to expanded storage is illustrated in Figure 5, Pages Per Sec To Expanded Storage. We note that transfers begin when there are approximately 100 logged-on users. This corresponds to the depletion of available central storage frames illustrated in Figure 4. This depletion would tend to decrease the amount of TSO logical swapping, causing more pages to be sent directly to expanded storage.

The availability of expanded storage frames decreases in concert with the increase in the pages per second sent to expanded storage. As discussed previously, migration is the mechanism used to regulate the number of available expanded storage page frames. Figure 6 illustrates the depletion of expanded storage frames as the number of active TSO users increases. As the number of logged-on TSO users increases, the number of swap sequences also increases leaving an ever increasing number of pages on expanded storage.

In Figure 6, we note a slight increase in the average number of available expanded storage frames with 190-200 average logged-on users. Based upon the measurement data, this is the point where the expanded storage frames are depleted to the extent that migration starts. Prior to reaching this level, we saw the depletion of central storage frames with 170-180 logged-on users. This caused a corresponding increase in the number of pages per second being sent to expanded storage. We can see the sequence of events occurring in MVS/XA's management of available storage resources and another cause and effect relationship in the MVS/XA operating environment.

As the number of available central storage frames decreases (Figure 4), we see an increase in pages per second sent to expanded storage (Figure 5) and a decrease in the available expanded storage frames (Figure 6). We now focus on the primary mechanism used by MVS/XA for managing expanded storage: migration.

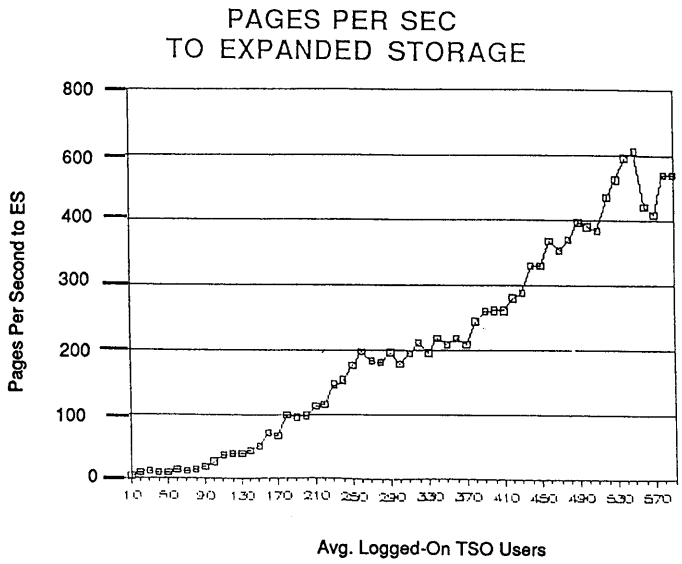


Figure 5

In Figure 7, we see the consequences of saturation on both central storage and expanded storage. We note no occurrence of migration until the number of logged-on users reaches 170-180. At this level we see a slight migration effort by MVS/XA which begins to replenish the expanded storage available frame queue. This is the Real Storage Manager's migration appetizer. The main course is served at the 200-210 logged-on user level and increases continuously until the system reaches 250-260 logged-on users. The next surge of migration activity occurs at 370-380 logged-on users. At this point, the average Migration Age will remain low indicating substantial contention for expanded storage. This results in continuous migration of expanded storage frames to auxiliary storage.

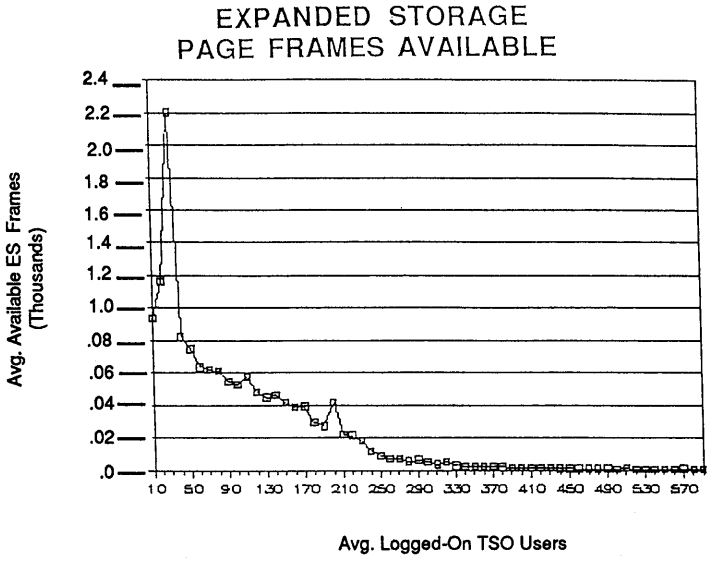


Figure 6

Another cause and effect relationship from MVS/XA memory management is shown in Figure 8. As MVS/XA begins page steal operations to replenish the central storage available frame queue and, subsequently, as migration activity begins to replenish the expanded storage available frame queue, use of the 3090-400 processor is affected. We see in Figure 8 a slight increase in processor utilization occuring at 170-180 logged-on users as MVS/XA begins migration operations to replenish the expanded storage available frame queue. The increase is due to extra processor overhead required by the migration process.

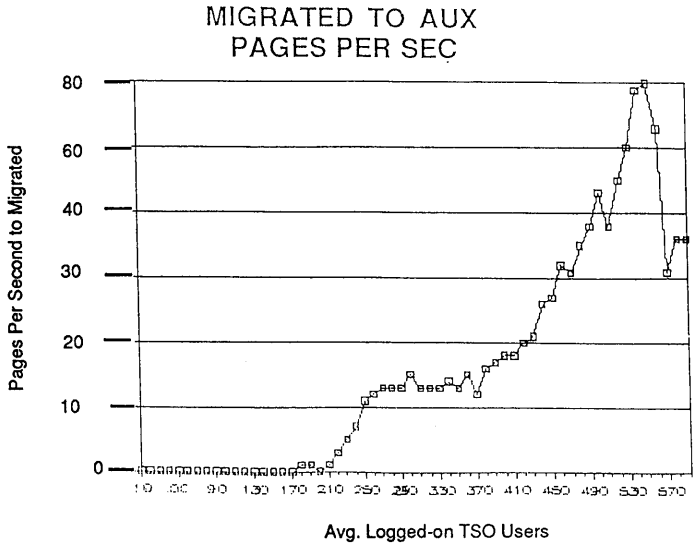


Figure 7

At 200-210 logged-on users, full scale migration of expanded storage pages to auxiliary storage begins. We again see an increase in processor utilization due to extra migration overhead. We can deduce from the graphs that MVS/XA page steal operations and expanded storage migration can have a negative impact on workload throughput and access to processor resources.

On some 3090-400 processors where the active number of address spaces exceeded 700-800, degradation was severe during MVS/XA page steal and migration operations. This problem was alleviated by APAR AY01085 which modified the MVS/XA page steal algorithms to replenish available page frames below the 16 megabyte line if there is a shortage in this region. This APAR has been superseded by PTF UY90024 and is available only for 3090-400 processors. We also see from the charts that once MVS/XA goes through the initial trauma of page steal and migration, these operations seem to smooth out as long as the workloads on the processor remain fairly stable. From Figure 8 we also see that the processor is not consistently saturated until we reach 490-500 logged-on users.

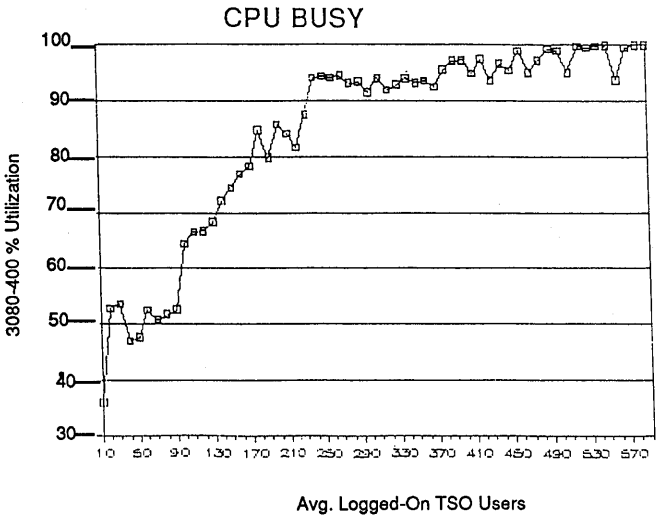


Figure 8

In Figure 9, TSO Throughput, we see that the transactions per second throughput rate does not appear to be adversely affected by MVS/XA page steal and migration operations on this processor. This has not been true on other 3090-400 processors that we have analyzed. We can surmise that the 'Large Systems Effect' is not as pronounced on processors where one type of work, such as TSO, is the predominant workload. Where there is a mixture of TSO and batch workloads on a

processor, batch requirements for central storage and expanded storage frames could have a substantial impact on the performance of the TSO workload. This would specifically be seen due to batch workloads having a greater demand for central storage page frames below the 16 megabyte line and an increase in MVS/XA page steal operations.

Though we did not see a significant impact on the TSO transaction per second throughput rate in Figure 9, the chart in Figure 10, TSO Response Time, provides more insight on the impact that MVS/XA page steal and expanded storage migration can have on a TSO workload's performance.

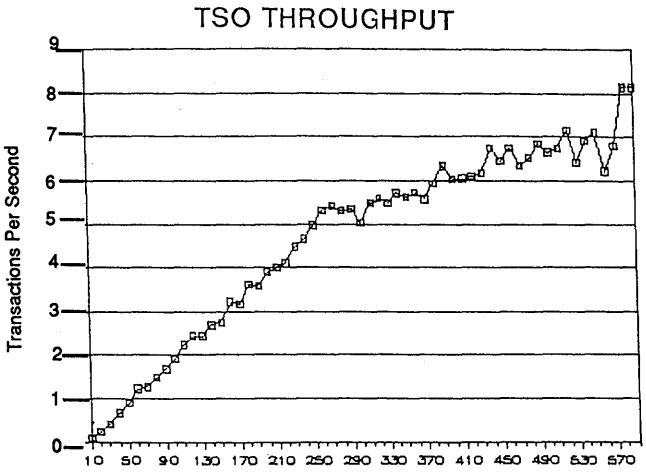


Figure 9

The sudden increases in the period one TSO transaction response times as shown in Figure 10 may be the result of MVS/XA page steal and migration operations, but viewed from a different vantage point. At lower numbers of logged-on users (less than 100), processor utilization and transaction throughput rates may not be as adversely affected by MVS/XA page steal and migration activities since any shortages of central or expanded storage would not be significant or prolonged with this smaller number of logged-on users. Also, LOGON activity will have a more pronounced effect on average response times with so few logged-on users.

Points of interest on the TSO Response Time chart are between 180-200 users and 210-240 users. At 180-200 logged-on users, we observed indications of MVS/XA page steal operations as central storage frames were depleted. At 210-230 logged-on users, we observed indications of expanded storage migration beginning to replenish expanded storage available frames. The TSO response times during these load levels reflect what could be the results of MVS/XA page steal and migration activities.

Consider next the 430-460 logged-on user level shown in Figure 10. The spikes in period one response times may be attributable to DASD delays (for user I/O) due to increased DASD activity from a larger number of users. Alternatively, they may be due to IPS MPL constraints. In either case, we would expect to see the trend continue and get progressively worse.

Clearly both MVS/XA page steal and expanded storage migration have many effects on throughput rates and response times. This is an area that needs a substantial amount of additional analysis. The 'Large Systems Effect' brought on by the increased number of active address spaces on the 3090-400 processor complex has revealed itself primarily in the area of storage management. With the announcement of the 3090-600E processor complex and the ability to operate with an even greater number of active address spaces, memory management and the demand for central storage frames below the 16 megabyte line may require extensive changes in the MVS/XA memory management algorithms.

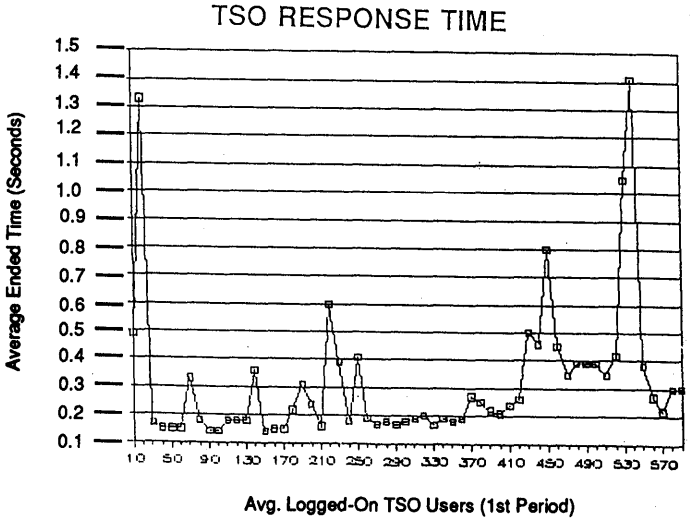


Figure 10

SUMMARY OF INTERACTION

We can summarize the interactions illustrated by the charts in Figures 4-10 with the following:

- Increase Number of TSO Users;
- Decrease Available Central Storage Frames;
- Trigger Movement of Page Frames to Expanded Storage;
- Reduce Available Expanded Storage Page Frames;
- Trigger Migration of Expanded Storage Frames to Auxiliary Storage; and
- Increase Processor Overhead to Support Migration.

The interactions have a net impact of:

- Throughput Rates Can Decrease;
- Response Times Can Increase.

The degree to which the net impact is felt will vary with memory constraints and task working set characteristics for the workloads being processed.

EXPANDED STORAGE CONSIDERATIONS

Expanded storage has helped alleviate many of the paging and swapping problems experienced by some installations.

Large TSO installations have received the greatest benefit from expanded storage due to the reduction in swap-in delay ('wait-to-start') time. Swapping is the primary mechanism for controlling TSO user access to resources, and swaps to expanded storage can be more than an order of magnitude faster than swaps to conventional auxiliary storage devices. This reduction in swap-delay time can provide improvements in both transaction throughputs and transaction response times.

TSO logical swap has been the previous mechanism for providing TSO period one users with required response times, often in the sub-second range, and for preventing swapping to auxiliary storage. With the availability of expanded storage, TSO logical swap should not be turned off (LSCTMTE = (0,0)) but may possibly be reduced. The

enhanced speed with which TSO swapping can occur through the use of expanded storage allows the freeing of central storage page frames (once used to hold logically swapped users) without suffering the swap-delays from auxiliary storage. It should also be noted that the 'think time grace period' for logically swapped TSO users was reduced from 15 seconds to 2 seconds in Release 2.1.3 of MVS/XA.

The Expanded Storage Criteria Table (ESCT) values in the IEAOPTxx member of SYS1.PARMLIB can be adjusted to bias the use of expanded storage for TSO, Batch or IMS/VS Virtual Fetch. The reader is cautioned that no guidelines have been made available by IBM for changing the ESCT constants or for providing a rationale for the default values illustrated earlier in this paper. Changes to the ESCT values should be made one at a time, and the effect of each change on workload throughputs and response times should be analyzed thoroughly before making additional changes.

The final benefit of expanded storage is a reduction in the utilization of channel paths and devices due to reduced paging and swapping activity to auxiliary storage. For many installations, this reduction has enabled installations to recover a significant amount of channel path, DASD controller and device capacity. Reducing paging and swapping device 'spread' has also allowed many installations to reconfigure their I/O subsystems and enhance overall I/O performance.

A note on paging data sets. Virtual storage backing for address spaces is still required in local page data sets until the availability of MVS/XA Release 2.2. Some installations have reduced the size of their local page data sets or eliminated local page data sets only to find that the virtual backing was still required and that page slot shortage messages are still issued by MVS/XA. Page slot shortages will prevent the creation of new address spaces.

CONCLUSIONS

Expanded storage can provide a substantial improvement in the overall performance of MVS systems, especially heavy TSO environments. However, as the number of TSO users increases, total available processor storage (central plus expanded) will eventually saturate and migration overhead can become a problem.

The IEAOPTxx parameters in the ESCT can, in principle, determine which classes of users suffer the most or least from migration overhead. In practice, substantial changes in these parameters produce only minor changes in relative performance. These parameters may, however, be of greater value in the future when additional facilities are introduced for the support of VSAM linear data sets, data in virtual, and special DB2 requirements.

REFERENCES

[1] Tucker, S.G., "The IBM 3090 System: An Overview," IBM Systems J., 25, 1, 1986, pp. 4-19.

A HEURISTIC MODELING APPROACH TO CICS CAPACITY PLANNING

Steven R. Hackenberg
Candle Corporation

ABSTRACT

Capacity planning for CICS has evolved from simple models and intuition to more sophisticated techniques which use a set of analytic models to calculate response time components of a transaction, such as network, device, and CPU times. Rarely do these models account for architectural limitations of CICS. This paper will focus on a simple model which considers the concept of software queuing as a starting point for projecting the effect of increases in current transaction volume and implementation of new applications.

INTRODUCTION

Consider the following scenario which will illustrate a common misunderstanding in data centers, and set the tone for the focus of this paper.

A large data center had 2 CPU's with workloads divided in the following way. System A was used to run TSO, production batch, and test batch. System B was used primarily to run the production CICS system with an insignificant amount of additional workload. System B was a 7.5 MIPS single-processor CPU running at less than 60% CPU utilization consistently during prime shift. Data center management was convinced that they could save a large amount of money by eliminating their excess capacity, so they swapped out System B and installed a 4.5 MIPS single-processor CPU. When the production CICS system was brought up during prime shift, response time had eroded from sub-second to 20 seconds, and during peak loads had become even worse. It became obvious that they had made a mistake but no one could understand the underlying reason. What they failed to consider was the simple concept of software queuing.

CICS ARCHITECTURE

To understand how software queing becomes an issue, a brief discussion of CICS architecture is appropriate. The primary purpose of CICS is to manage resources, such as terminals, files and storage, while performing services on behalf of transactions. Another major function of CICS is to control the execution and serialization of transactions as they request CICS services and execute application programs. The current status of each transaction at any instant in time will be indicated in the Dispatch Control Area (DCA), a control block created to mark the existence of a unit of work. All the services performed by CICS are divided between MVS (or DOS) tasks as illustrated in figure 1. Some services performed, such as a request for record journaling, are executed under a separate Task Control Block (TCB). A key point is that all serialization activities and application program execution is performed under the umbrella of a single TCB, shown in figure 1 as the on-line TCB. The outcome of this architecture is that the bulk of the work performed by CICS, thus the vast majority of the CPU consumed by CICS, is done by a single server element. This leads naturally to the potential for software queuing.

PREDICTING SOFTWARE QUEUING

Understanding the degree of software queuing in the context of CICS can be expressed in the following way. What is the probability that at any instant in time, more than one transaction, or DCA, will be in a ready status. To further define this concept, let's look at it from an operating system viewpoint. The CICS workload either has work to be performed (ready), or it does not (not ready). Given the availability of the CPU resources, CICS in a ready state, and the lack of a higher priority workload (i.e. operating system page handling, VTAM, etc.) CICS will be given control. Any attempt to monitor CICS from an operating system standpoint will only show the availability of the CPU resource, and CICS's ability to exploit it. This leaves three possible conditions: 1) CICS ready (or actually executing); 2) not ready; 3) ready, but unable to run because of pre-emption by a higher priority workload (waiting for CPU). The capability to detect software queuing within a CICS environment does not exist within an operating system monitor because it can only see CICS's ability to exploit the CPU resource, but not to what theoretical depth.

Just as competing workloads being controlled by an operating system can wait on CPU, transactions competing for machine cycles on a dispatch chain within CICS can be hindered by other transactions. Having one CICS transaction on a DCA chain in a ready status, all succeeding ready transactions can be thought of as waiting on dispatch. The reason that Wait-On-Dispatch is considered a software queue is because the architectural limitations of a single CICS address space precludes the ability to make effective use of a multi-engine CPU complex such as a 3090 model 400.

The actual prediction of Wait-On-Dispatch is directly related to the aggregate consumption of CPU cycles by the CICS workload itself and all higher priority workloads. Experience has shown that typically when total CPU consumption of CICS, and higher workloads, remain at less than 40%, the amount of Wait-On-Dispatch is negligible. Once this total workload begins to exceed 50% CPU utilization, Wait-On-Dispatch becomes significant and in fact accelerates rapidly. The degree of acceleration is dependant on the capacity of the CPU as shown in figure 2. This diagram is for illustrative purposes only. The actual point of acceleration will be dependant on factors such as the profiles of the transactions running on the particular CICS system. For example, a CICS system which has short application program path lengths with a high number of file accesses, and VSAM subtasking in effect on a dyadic processor, may move the acceleration point beyond the expected norm.

DETECTING WAIT-ON-DISPATCH

Most CICS monitors in existence today do not detect Wait-On-Dispatch. They fail to see this condition because they are event driven, typically from the CICS trace facility. To illustrate, let's consider a transaction in execution fairly low on an active DCA chain which requests a file access. A trace record is created indicating the call to File Control and the task is marked not ready. When the File Control request is completed, CICS will mark the DCA for the task as ready to run on the next DCA search, but no trace record will actually be created until the task is re-dispatched. In an over committed CICS environment, the length of time between completion of a file control request and a re-dispatch could easily be more than half of the total transaction response time.

The most practical way to detect Wait-On-Dispatch is by using a sampling technique where the active DCA chain is interrogated at fixed time intervals. An interval of 2 seconds works well to accumulate counts for total active tasks and task queuing. Because of the asychronous processing design of CICS, the DCA will not be updated until DCA search time; therefore, the analyzer will have to determine if the ECB associated with the DCA has been posted complete. The data should be logged for later analysis, and the accumulators should be zeroed out on a periodic basis, such as every 30 minutes, so that the degree of Wait-On-Dispatch can be measured for various times of the day. If you do not have a monitor which will report Wait-On-Dispatch statistics you can write one with some effort. The accumulated numbers should be put into the following formula:

$$\frac{\text{Total Ready Tasks After First}}{\text{Total tasks found on Active Chain}} = \text{Wait-On-Dispatch as Percentage of Response Time}$$

In the absence of a monitor which is sensitive to the existence of Wait-On-Dispatch, users can predict, with some accuracy, the acceleration point based on intuition. The simple fact is that as CICS and higher priority workloads begin to move beyond 50% total CPU utilization, the probability of software queuing is large, and the rate of acceleration of degredation due to Wait-On-Dispatch will increase as aggregate CPU consumption continues to increase.

Going back to the original case study, additional conclusions can be drawn from our scenario. Even though total CPU consumption was at 60%, about 50% of it was attributable to CICS and MVS related overhead. Software queuing had probably started to occur significantly. By down sizing the CPU from 7.5 MIPS to 4.5 MIPS, they increased aggregate CPU to above 75%. Had they fully understood Wait-On-Dispatch, they would have known that response time would suffer drastically in this situation. As it turns out, the monitored Wait-On-Dispatch was consistently greater than 68% of response time, and the execution phase of response time was running at about 30%, a significant increase due to the slower CPU speed.

MOVING THE ACCELERATION POINT

In a multi-engine CPU complex, there is a simple technique which can be used to move the acceleration point and thus devote more CPU capacity to CICS without degrading response time. The solution is simple--make use of the MRO capability offered by CICS. To reiterate our previous discussion, the primary cause of software queuing at such a low CPU level is due to the single server element restriction in the CICS architecture. The easiest way to solve this problem is to create a multi-server element environment for a given CICS workload. Let's consider an ideal situation. We have a 3084 quadratic CPU complex. We create an MRO environment with a Terminal Owning Region (TOR) and three Application Owning Regions (AOR's), dividing our application volume equally between the three AOR's. What we have accomplished is that we now have four servers for the same CICS workload. To further define our ideal situation, we have created four online TCB's controlling four DCA chains, all of which can be simultaneously dispatched on the four engines of our 3084 CPU complex. Theoretically we have created an environment which will allow CICS to exploit all 100% of our 3084 with no appreciable degradation due to Wait-On-Dispatch.

Now let's set aside our "ideal" and consider reality. The TOR itself will consume far less than 1/4 total CPU. Its basic function will be to read screens and hand off transactions to the AOR's, and receive and redisplay screens from the AOR's. In a large CICS network, chances are good that the TOR will occasionally have multiple screens to process at the same time and, as a result, become a bottleneck. Also, MRO does not come for free. There will be a certain amount of overhead associated with managing the shifting of transactions to other address spaces, and the potential exists for an increase in real storage commitment. Using this approach, the acceleration point for Wait-On-Dispatch before a noticeable erosion in response time is probably above 70% total CPU. At the very least, MRO in a multi-engine CPU complex offers some flexibility.

CONSIDERATIONS MOST OFTEN FORGOTTEN

Certain pitfalls exist that are easily forgotten which can effect the amount of Wait-On-Dispatch experienced. Mentioning a few of these will give users a feel for what to look for when response time degrades due to software queuing when there is no appreciable increase in CICS volume.

VM

Running a CICS system under MVS (or DOS) under VM will almost guarantee a 20% Wait-On-Dispatch. (The 20% figure is the most common one seen in the field.) This really makes sense when you think about it. VM itself will carry 10-20% overhead just by the nature of what it does. Next, you have an operating system and related overhead taking 15-20%. The presence of VM will cut down on the amount of available real storage causing even more overhead. The net result is that we already have about 40% of a CPU being consumed above the CICS workload, and we have not yet considered the CPU being consumed by CICS itself. Even with a moderate CICS system taking 15% of a machine, we have probably gone past the point were software queuing becomes a problem.

PAGING

Many people have fallen prey to Wait-On-Dispatch because of total system wide paging. Users can assure that CICS itself does very little paging by using storage isolation. But, if total paging in the system gets out of hand the operating system overhead to process damand paging will begin to drain valuable CPU availability by increasing aggregate CPU consumption above the CICS workload.

VTAM, JES2, ETC.

Although typically the service address spaces, such as GRS, VTAM, JES2, CATALOG, etc., consume very little CPU, there are occasions when this becomes a problem. In this case users should consider moving JES2 below the CICS workload in priority but, the point is to make sure you understand the total CPU being consumed by these service processors.

CLASS MAX TASK (CMXTSK)

Due to application design limitations, or other reasons, users sometimes have a requirement to use CMXTSK to control certain applications. Prior to CICS 1.7, CICS attempts a conditional attach of any transaction which has been classed so that task control can assure that the class limitation can be honored. CICS will re-attempt the conditional attach for transactions deferred due to Class Max, based on the Scan Delay value of 0 milliseconds. In some cases heavy use of CMXTSK has caused over 10% CPU to be consumed just because of a high number of attempts to attach. This problem is easily solved by setting scan delay to 500 milliseconds or more.

SUMMARY

Many data centers have fallen into a false sense of security when considering the capacity requirements to support a CICS system. How many of us have heard the cry "Hey, no problem. Bring on that new application, we have PLENTY of CPU." A large number of people just do not realize that long before CICS can exploit 100% of a CPU, software queuing will potentially degrade response time far beyond acceptable service levels.

The first step to take is to understand the current CICS workload and the CPU utilization of all higher priority workloads. With an awareness of software queuing and the point at which it will most probably begin to become a serious problem, users will be in a far better position to accurately predict their CPU needs. This heuristic approach has the advantage of being based on the actual environment without the need to make assumptions, and it is an approach which everyone can easily understand.

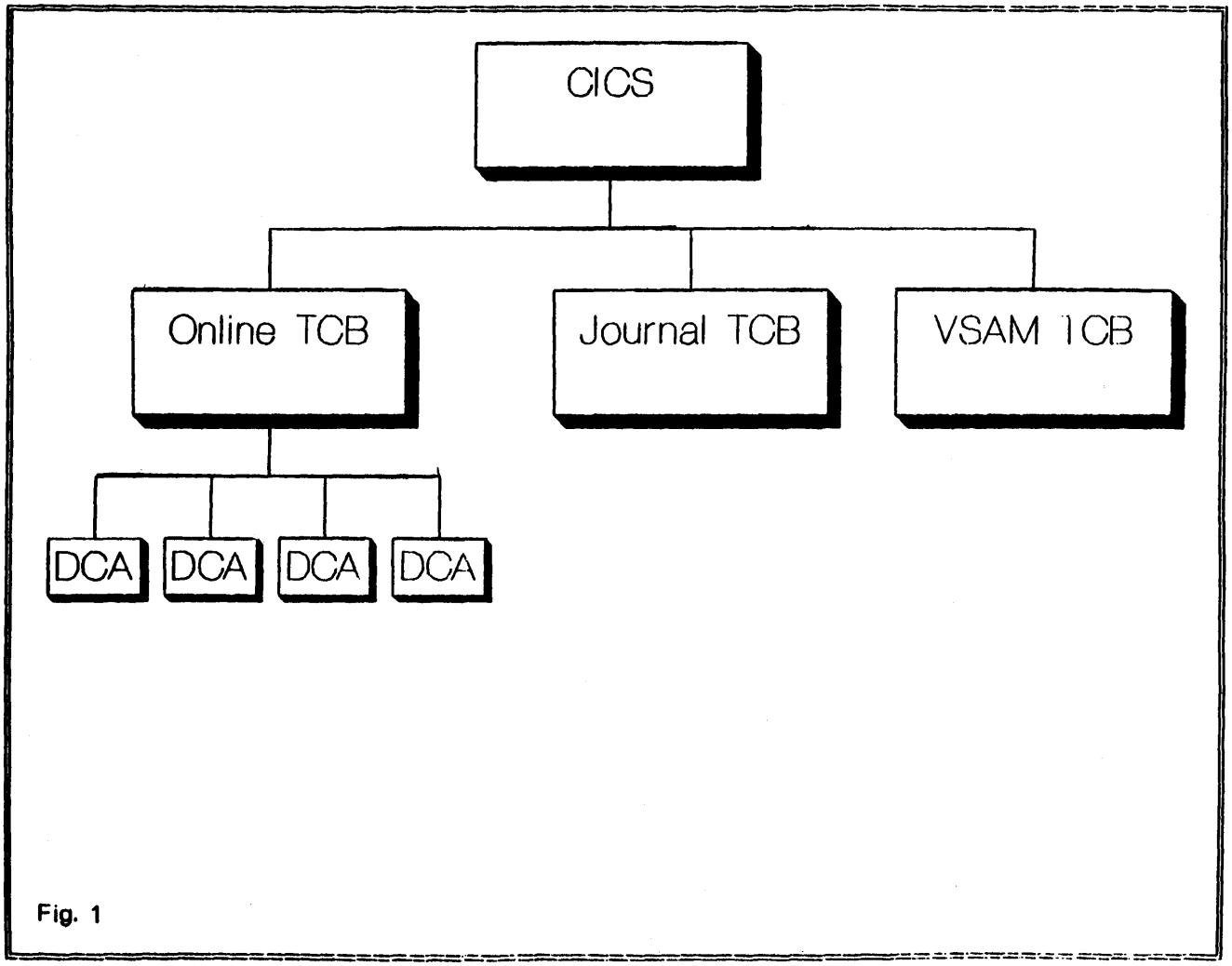


Fig. 1

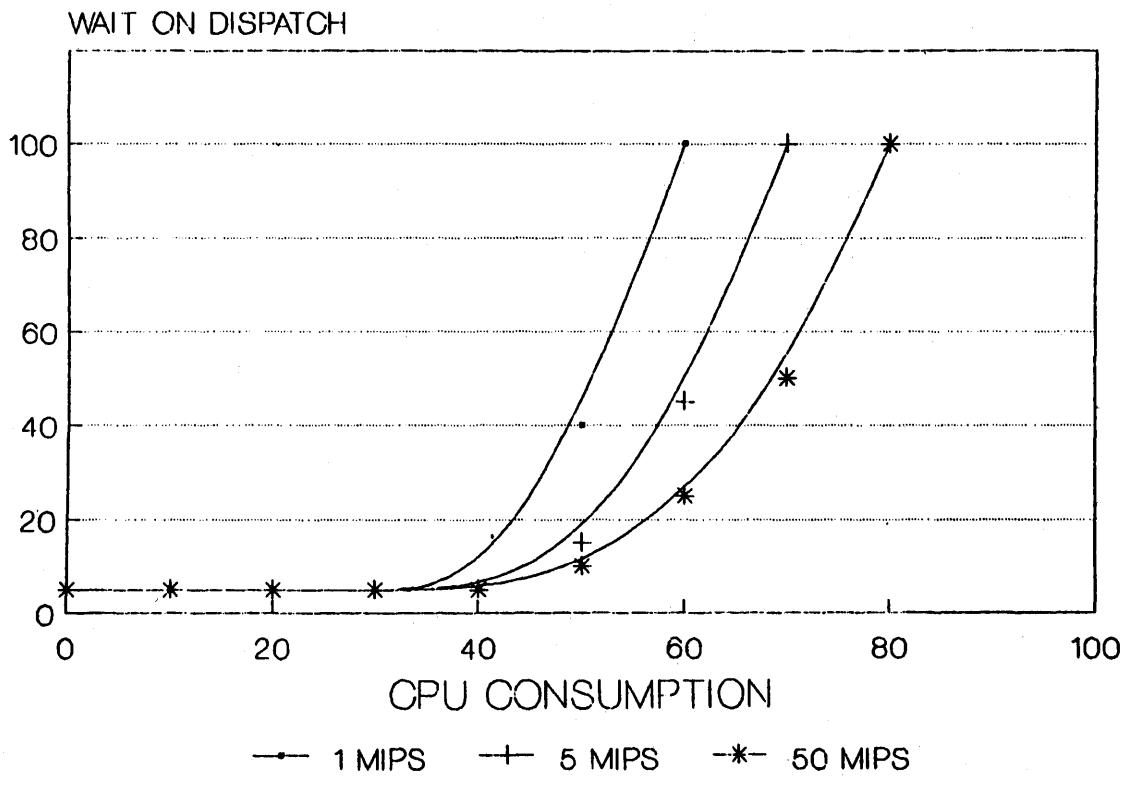

WAIT ON DISPATCH
100
80
60
40
20
0
0
20
40
60
80
100
CPU CONSUMPTION
1 MIPS
5 MIPS
50 MIPS

Fig. 2

PAGING ANALYSIS IN AN EXPANDED STORAGE ENVIRONMENT.

Thomas Beretvas
International Business Machines Corporation,[1]
P.O. Box 390, Poughkeepsie, NY 12602

ABSTRACT

Multiple Virtual Storage (MVS) processor storage management and paging algorithms are reexamined with an emphasis on 3090 environments with expanded storage. Page data set space allocation recommendations are given. Paging device performance characteristics are discussed and configuration guidance given. The state transitions to various swap states are defined and illustrated. A case study of actual RMF data is performed for a 3090 environment.

Methods are described to identify and solve paging problems.

INTRODUCTION

Multiple Virtual Storage (MVS) processor storage management and paging algorithms are reexamined with an emphasis on 3090 environments with expanded storage.

A previous publication [1] gave MVS/370 paging guidelines. This paper assumes the use of the MVS operating system using extended architecture (MVS/XA). MVS/XA paging guidelines were presented before in [2]. This paper further assumes the use of IBM 3380 paging devices with Device Level Selection. The emphasis of the paper is on IBM 3090 processor environments, with expanded storage, unless otherwise stated.

Page data set space allocation recommendations are given. Paging device performance characteristics are discussed and this results in an easy-to-use methodology, sometimes known as "rules of thumb" for establishing paging configurations.

The state transitions to various swap states are defined and illustrated. A case study of actual RMF data is performed for a 3090 environment.

Methods are described to identify and solve paging problems.

Why consider paging at all?

In the last few years there has been a growing emphasis on interactive response time. Fast, subsecond response time has become a key to productivity [3]. Analysis of MVS and VM operating system environments revealed that--in the past--paging delays represented a major component of the response time, especially when the response time was not very good. Often, as high as 60 percent of the response time for trivial transactions was attributable to paging delays.

Paging, as the key problem is rapidly disappearing for installations with adequate processor storage configurations. This is particularly true for IBM 3090 installations with expanded storage. Even for these installations there is still a need to size an adequate paging configuration. The paper provides guidelines for this task.

PROCESSOR STORAGE MANAGEMENT

The management of processor storage (i.e., central storage and expanded storage) must be understood if paging is to be understood. MVS manages processor storage in fixed size (four kilobytes) pages. In MVS multiple "concurrently" running programs contend for the same limited processor storage space. If expense did not matter, at least conceptually, enough processor storage could be provided to accommodate all competing programs. But money does matter, thus processor storage is sometimes limited in size even today.

In a system with only central (previously known as main) storage "competing" programs share this storage. Through the years paging strategies have evolved, all centering on keeping the "needed" pages in central storage while moving the unneeded pages onto direct access storage devices (DASD), which act as paging devices. The reason for this strategy was that megabyte for megabyte, DASD storage space is much cheaper than central storage, but if "unneeded" pages are suitably selected and moved to DASD, then central storage occupancy is significantly reduced, and the delay caused by pages retrieved from DASD does not occur frequently.

In a system with expanded storage unneeded pages from central storage are usually first moved to expanded storage; then unneeded pages from expanded store are migrated to DASD via central storage. Of course, any page that is referenced by a program, must be retrieved from expanded storage or DASD into central storage. Page retrieval from DASD may cause quite a long delay (20-30 ms) for the program; from expanded storage the delay would be much shorter, about 75 microseconds.

---

[1] Permission is granted to CMG to publish this paper. IBM retains its right to publish this paper elsewhere and distribute copies of it to whomever it chooses.

PAGING STRATEGIES

The paging strategies attempt to keep the "needed" pages in processor storage while moving the unneeded pages onto auxiliary storage devices (DASD).

The paging strategies are based on three basic principles:

- If a user becomes **inactive** (e.g., goes into terminal wait status), the system assumes that his pages are unlikely to be referenced in the near future. Thus, if there is a shortage of frames, all (or most) of the user's pages are removed from processor storage to DASD. This is called a physical swap-out. Swap-outs free many processor storage frames. This is the primary mechanism for controlling processor storage occupancy.

- MVS attempts to satisfy the processor storage frame needs of all executing address spaces by keeping their working sets [4] (loosely: the pages that are required for execution) in processor storage. Once this need is satisfied, MVS stops short of completing swap-outs, and does "processor storage swaps;" using **excess processor storage** to retain the "swap group" pages for idle address spaces. The swap-out process includes defining the set of pages most likely to be needed the next time the address space executes. This is called the "swap group," and it is an approximation of Denning's "working set".

  Retaining the swap group means that pages not belonging to the swap group receive less favorable treatment. These pages are "trimmed". Page trimming is the second, most frequently used method for freeing up processor storage.

- It is assumed that if an active user does not reference a page for a long time (e.g., 20 seconds) then the page is not required in central storage. From this, we can infer that the page will not be required in the near future, and can be "paged out", [2] and the processor storage frame containing it is thereby freed. Page-outs are therefore selective and are based on the length of time that a particular page has been unreferenced. Only a few of the pages belonging to a user are removed. Similar strategies are used for both central and expanded storage.

Processor storage swapping reduces the activity to the auxiliary storage devices and improves response time by eliminating the I/O delay that would be necessary to (physically) swap-in the address space when it is again ready to execute. The elimination of swap-in delay is often critical to meet response time objectives for highly interactive workloads such as TSO and CADAM.

"Processor storage swapping" is an extension of what used to be called logical swapping. **Logical swapping** retains the inactive user's pages in **central** storage. A shortage of central storage frames may force these pages to be moved into expanded storage, causing an **expanded storage swap**. Processor storage swap is therefore a generic name encompassing both logical and expanded storage swaps.

In converting a logical swap into an expanded storage swap, the system distinguishes between "primary" and "secondary" working sets. The primary working set contains fixed and local system queue area (LSQA) pages and the first page of each virtual segment; while the secondary working set is the remainder. (Analogously to the **two-stage swap** prior to the advent of extended swap in MVS release MVS/SP 1.3.0). The distinction between primary and secondary working sets is important. If there is a shortage of expanded storage frames, then these pages in expanded storage may **migrate** to auxiliary storage (DASD) via central storage. During and after the migration, the primary and secondary working sets **retain** their identity and are handled differently.

[2] This is called least recently used, or LRU strategy, and it is seldom invoked in a well tuned installation.

PAGING OPERATIONS

Until now we discussed processor storage management. If processor storage is insufficient to contain all pages required, then paging to auxiliary storage occurs. In the following discussion, it is assumed that paging DOES occur. Please note however, that with increased central storage sizes and virtually unlimited expanded storage sizes, paging should not be a problem in well-configured modern installations.

Four different paging operations to/from auxiliary storage are distinguished:

1. **Page-ins.** When a program is referencing a page which is not in central storage, it page faults. If the page resides on auxiliary storage, an asynchronous page fault occurs; prompting the operating system to retrieve the page. that page by itself, from DASD. While the page fault is resolved, the program waits until the page is obtained from DASD. During this time, other programs may execute. Note that while page-outs usually involve multiple pages being written to DASD, page-ins (page faults) usually result in single page reads.

   If the page fault references a page in expanded storage, this is resolved synchronously, i.e., the page is retrieved from expanded store without surrendering control of the processor to another program.

2. **Page-outs.** The obverse of page-ins are page-outs. It is likely that a paged-out page will be subsequently page-faulted back into central storage.

   In installations with ample expanded storage, page-outs to expanded storage occur in preference to DASD, to exploit the faster page fault resolution of expanded storage.

   In environments with constrained or no expanded storage, page-outs to DASD occur. To better use the paging DASD resources, page-outs usually cause multiple pages to be written out at one time. There are three different kinds of page-outs: page steals, swap trims, and migration page-outs.

   - Page steals. An example of "page stealing" is when old (long time unreferenced) pages belonging to active users are taken. This tends to occur only in case of central storage shortage. A maximum of 10 pages are stolen from a user at one time.

   - Swap trim. A typical example of swap trim is when **unreferenced** pages belonging to users who go from active to inactive state are taken. Trimming is not limited in size and may involve many pages of a user. The primary working set contains referenced pages, thus these pages are not trimmed.

   - Migration page-outs. Migration page-outs occur because the system "migrates" non-working set i.e., stolen or trimmed pages from expanded storage to auxiliary storage via central storage.

The page-out blocking factor to DASD is determined by the auxiliary storage manager (ASM), and it is set to a maximum of 30 for 3380s, i.e., up to 30 pages can be blocked in a channel program. Migrated pages and swap trimmed pages may be paged out with a blocking factor of up to 30. The stolen pages have a maximum blocking factor of 10, since the limit of one time page steals from an address space is 10. The number of pages actually blocked is a function of the number of pages actually taken by the real storage manager (RSM). Thus, e.g., assume that 44 pages were trimmed for an address space. Then two blocks of pages are created; one of 30 pages and another one of 14 pages.

Note that the algorithm attempting to obtain contiguous slots on auxiliary storage is NOT invoked for page steals, and migration page-outs. This means that while the central processing unit (CPU) overhead of multiple start subchannel operations and interrupt handling is avoided, the DASD write operation is not fully optimized (intentionally). The non-contiguous writes are required to avoid fragmentation of the disk space.

3. **Swap-outs.** Many users of these systems are interactive, whose typical activity pattern is a short interaction with the system, followed by long inactivity or think time. As an example, interactions of 300 ms duration are common which are followed by inactivity lasting from five seconds to several hours. For these long thinking users, it makes good sense to get rid of all their currently held pages; this is the physical "swap out." Pages selected for swap-out are pages of the swap group.

   Given ample expanded storage, swap-outs often occur to expanded storage, because the swap-in speed from expanded storage is about two order of magnitudes faster than from DASD. (Noting that the two kinds of swap-in operations differ significantly, one may get an idea of performance difference by observing that a swap-in of 12 pages from DASD may take 80 ms, while a swap-in of 12 pages from expanded storage may take less than 1 ms). If the available expanded storage is insufficient to accommodate swap-outs, then the swapped out pages are written to DASD. There is a general control mechanism which determines whether the the address space is important enough to be swapped out to expanded storage or DASD. The decision depends on the quantity of expanded storage available, or more specifically on the "migration age" (briefly described later).

   A set of swap pages written to an actuator by a single SSCH instruction is called a swap set. A swap-out to DASD usually results in the swap sets being written into contiguous set of slots. The rationale for this kind of allocation is that a successful request results in a single seek and no latency between pages, thereby minimizing device service time and utilization.

   When swap data sets are used for swap-outs, the swap data sets are preallocated into contiguous sets of slots ready to receive the swap sets. Thus, in this case the swap sets are always written contiguously by the inherent nature of swap data sets.

   When local page data sets are used for swap-outs, the situation is more complex. The ASM "contiguous slot allocation" algorithm attempts to find a set of contiguous slots on the data set. The search for contiguous slots can be quite costly in CPU time however. Simulation results indicated that this search should not be undertaken unless there is an abundance of unused slots on the device. As a rule of thumb it might be considered that the number of slots available on the device should be at least four times as high as the number of occupied slots, for this algorithm to be operational.

   In a system with expanded storage, most swap-outs to DASD are migration swaps i.e., the user's working set is first logically swapped, then the logical swap is converted into expanded storage swap, and last, the primary and secondary working sets are--separately--migrated to auxiliary storage via central storage. If swap data sets are used, all direct auxiliary swaps and migrated primary (but not secondary) working sets go to them. If swap data sets are not used, all swaps go to local page data sets.

4. **Swap-ins.** Swapped out pages are retrieved by swap-ins. Swap-ins from expanded storage result in getting the pages in the primary working set, and subsequently page-faulting on the pages in the secondary working set, as required.

   A swap-in from auxiliary storage results in a retrieval of all the swapped out pages i.e., the previously written swap group of swap sets for this address space is brought back into processor storage. The system attempts to read the entire swap group of a physically swapped user **all at once**. A two-stage swap takes place for those users whose physical swap occurred because of migration from expanded storage. Primary working set is swapped in first, followed by the swap-in of secondary working set pages. The secondary working set usually consists of several swap sets.

   The swap-in of all the individual swap sets in a swap group or a secondary working set will take place concurrently, if these swap sets are located on distinct data sets, devices and paths. The concurrent I/O operations reduces the total time taken for swap-in, and exploits the bandwidth of multiple paths.

CHARACTERIZING PAGING OPERATIONS

PAGING TYPE	PAGES/ IO REQ.	PAGE DATA SET TYPE
PAGE-IN (FAULT)	1	LOCAL
STEAL (PAGE-OUT)	1-10	LOCAL
VIO	1-10	LOCAL
SWAP-OUT/IN, MIGRATED PAGE-OUTS, SWAP TRIM, MIGRATED SECONDARY WORKING SET	1-30	LOCAL
SWAP-OUT/IN, MIGRATED PRIMARY WORKING SET	1-12	SWAP
PLPA	1	PLPA
COMMON	1-10	COMMON

Figure 1. PAGING TYPES. (MVS)

Quantitative characterization of the various ASM paging operations is summarized in Figure 1. Page faults usually result in a single page transferred from DASD to real storage. Page-outs usually cause multiple (up to 30) pages transferred to local page data sets in a single I/O operation.

Swapping may occur to local or swap data sets. Swapping to swap data sets occurs in swap sets which may contain up to 12 pages. Since the use of swap data sets is usually inefficient in 3090 environments, the use of swap data sets will not be discussed in any detail. Swapping to local page data sets occurs in swap sets containing up to 30 pages. Local page data sets are not preallocated into swap sets since demand paging also occurs to these data sets, but the contiguous slot allocation algorithm attempts to place pages to be **swapped** into contiguous slots.

Paging from pageable link pack area (PLPA) and from/to COMMON page data sets is not discussed in this paper, since most of what can be said about demand paging considerations from LOCAL page data sets is directly applicable to PLPA and COMMMON page data sets.

The system interactive response time is directly influenced by the page-in and swap-in response times; page-out and swap-out response times do not directly impact system response times.

A PAGING CONFIGURATION

SWAPPING AND PAGING TO ALL LOCAL PAGE DATA SETS

In an expanded storage environment migration swaps go to local page data sets only. Hence in this environment usually only the use of all local page data sets make sense. These data sets can block up to 30 pages per I/O when used for swapping. The local data sets also handle the demand paging.

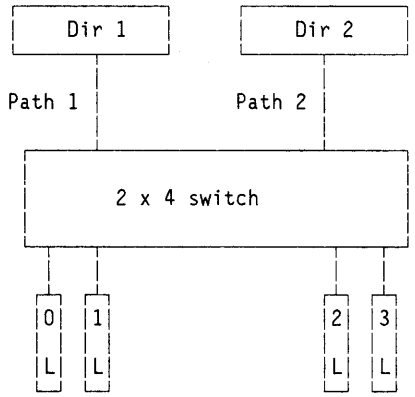


Figure 2. ALL LOCAL PAGING PATHS

Figure 2 shows all local paging paths, with four local paging actuators on two paths, with all actuators doing both paging and swapping. Assume that swapping occurs with 30 pages per I/O, page-outs with five pages per I/O and page-ins with one page per I/O.

Assume a swap to demand paging ratio of 4:1 with a swap load of 100 pages per second and a demand paging load of 25 pages per second per path.

		Page-in	Page-out	Swap
Blocking factor	=	1.0	5.0	30.0
Paging rate per actuator	=	6.25	6.25	50.0
Total path paging rate	=	125		
Path utilization	=	25.0 percent		
Actuator utilization	=	26.8 percent		
Actuator response time ms	=	28.0	33.5	76.5

Figure 3. LOCAL PAGING DEVICES WITH DEMAND AND SWAP PAGING

Figure 3 presents modelling results for all local page data sets handling both paging and swapping traffic. Observe that page-in response time is below 30 ms, which is my "rule of thumb" for acceptable response time. This rule of thumb was established pragmatically, it showed good overall response times, and did not result in excessive paging delays. Note the following advantages for the "all local" configuration.

1. ASM load balancing selects the local page data set with performance that is better than average among the set of local page data sets. Thus, local paging data sets on non-dedicated actuators or heavily loaded paths encounter lower paging loads. The automatic load balancing therefore corrects some performance problems dynamically. This capability exists only for local page data sets.

2. Migration paging from expanded storage is largely directed to local page data sets. If swap data sets are defined, only primary working sets are migrated to them. Thus, in an expanded storage environment the use of swap data sets is STRONGLY discouraged, because they tend to be underutilized.

UTILIZATION LIMITS FOR LOCAL PAGE DATA SETS

Modeling calculations have been used to establish device use limits (i.e., conditions under which the device utilizations and response times are reasonable and the queues occurring are not excessive).

Somewhat arbitrarily these limits were defined in such a way that paging path and device utilizations were kept below 35-40 percent. Experience shows that these are reasonable guidelines. Since the guidelines are somewhat arbitrary, an installation can load its devices more than the guidelines indicate, if it is willing to tolerate worse response times.

Device use limit tables for 3380 local paging devices are built with these guidelines. Three separate cases are considered:

- Dedicated demand paging paths (i.e., no swapping)
- Dedicated paging paths (with swapping)
- Nondedicated paging paths

LIMITS ON DEDICATED DEMAND PAGING PATHS

Number of actuators per path	4
Paging rate per actuator	< 25
Response time ms	< 30

Figure 4. LIMITS ON DEDICATED DEMAND PAGING PATHS

Figure 4 contains guidelines for dedicated demand paging paths. In such an environment it is assumed that swapping does not go to these paths, either because ASM swapping does not exist or it is minimal.

The guideline states that the maximum paging load for a 3380 paging actuator in a dedicated demand paging path should be 25 pages per second. This would result probably in a response time which is better than 30 ms. There should be at most four such demand paging actuators per path.

LIMITS ON DEDICATED PAGING PATHS (WITH SWAPPING)

Number of actuators per path	2
Swap to demand page ratio	>4:1
Paging rate per path	<125
Paging rate per actuator	< 63
Demand paging response time ms	< 30

Figure 5. LIMITS ON DEDICATED PAGING PATHS (WITH SWAPPING)

Figure 5 contains limits for the case of dedicated all-local paging paths that have both paging and swapping.

The essence of the guideline is that a local page actuator is required for:

- every 100 swap pages per second; and
- every 25 demand pages per second.

Hence, assuming a swap to paging ratio of greater than 4:1, two local paging data sets on a path can each sustain overall paging rates of up to 63 (since 2 times 63 equals 126) pages per second.

The path and actuator utilizations should be kept below 35-40 percent, so that the demand paging response time can be kept below 30 ms.

NONDEDICATED PAGING PATHS

Paging type on path	Demand only	Page and swap
Number of PAGING actuators per path	< 4	1
Number of actuators per path	4	4
Paging rate per paging actuator	< 25	< 63
Demand paging response time ms	< 30	< 30

Figure 6. LIMITS ON PATH WITH DATA AND PAGING

Figure 6 summarizes some general rules for the case when (nonpaging) data sets containing data are commingled on the same path with local page data sets. The guidelines basically repeat the guidelines presented earlier. In general, this is a cost-effective solution, if the rules (on the maximum paging rates and number of paging actuators per path, and path utilization) are followed. Only one local page data set **doing swapping** (TSO environment) is allowed per path, but up to 4 demand paging data sets (DB/DC environment) are allowed. In order to limit the impact of mixing paging and data both, it is wise to limit the **overall** path utilization to a value below 35 percent.

For good performance one should place AT MOST two actuators that may service swaps into a string-switched complex of two paths.

CONFIGURING A SYSTEM

GUIDANCE FOR PAGING CONFIGURATION MIX

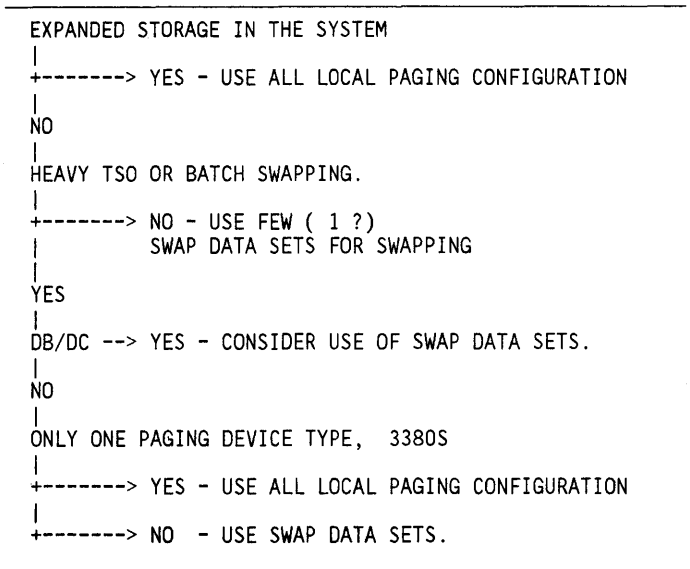

EXPANDED STORAGE IN THE SYSTEM
YES - USE ALL LOCAL PAGING CONFIGURATION
NO
HEAVY TSO OR BATCH SWAPPING.
NO - USE FEW (1 ?)
SWAP DATA SETS FOR SWAPPING
YES
DB/DC --> YES - CONSIDER USE OF SWAP DATA SETS.
NO
ONLY ONE PAGING DEVICE TYPE, 3380S
YES - USE ALL LOCAL PAGING CONFIGURATION
NO - USE SWAP DATA SETS.

Figure 7. DETERMINING PAGING CONFIGURATION MIX.

Usually the first question raised in configuring a system is whether swap data sets should be used. The easy and general answer is "NO" for 3090 environments with expanded storage. Figure 7 contains a guide in flowchart form with more detailed specific guidance on whether swap data sets should be used or not.

If there is expanded storage in the system, the use of swap data sets makes very little sense.

If there is no expanded storage in the system, and there is a swapping environment together with the presence of DB/DC, which is sensitive to demand paging, swap data set use should be considered.

If only one paging device type is used (especially 3380s) "all local" page data sets should be used most of the time.

In non-TSO environments a few (probably only one) swap data sets could be used to accommodate an occasional burst of heavy batch swapping and to avoid interference on the demand paging actuators.

PAGING SPACE CONSIDERATIONS

An important issue is the size of local page data sets. MVS currently (MVS/SP Release 2.1.3) limits the size of a 3380 page data set to 65400 slots. (That is about 250 megabytes). The sum of all local page data set space allocated should be at least **twice** as high in a 3090 environment as the available processor storage. Thus, if a 3090 has 128 MB of central and 256 MB of expanded storage for a total 384 MB of processor storage, then there should be at least 768 MB of paging space, (i.e., one can just about manage with three actuators). This rule of thumb ignores actual paging rates.

If any actual paging DOES occur, local paging actuators will be used more efficiently, because the contiguous slot allocation algorithm is used, if the allocated space on an actuator exceeds the space used by a factor of more than about four. Hence, in the above example, one should have at least six paging actuators allocated to the maximum extent permitted. In fact, I would modify this by saying that one needs about eight paging actuators. (The number is inflated to be on the safe side. In a knowledgeable installation the excessive allocation may not be required). In other words, there should be one 3380 local paging data set of 65400 slots on each of eight paths.

However, in most cases the actual paging load will be insignificant (i.e., a total of around 100 pages per second). Thus, the use of eight very lightly used dedicated actuators is not very cost-effective from the installation's point of view.

Thus, my new recommendation is to consider **the use of nondedicated paging actuators** on nondedicated paging paths. A page data set uses about half of a 3380 model D actuator, and less than one quarter of a model E; so perhaps model E use should be considered. The only concern is to keep both the actuator and path utilizations below 30 percent. This says that the use of expanded storage frees up (at least partially) the previously dedicated paging actuators for other use.

-Allocate 65400 slots per local page data set

-Use at least eight local page data sets

-Use nondedicated paging devices

Figure 8. PAGING SPACE CONSIDERATIONS

Figure 8 summarizes the considerations just discussed.

A CONFIGURING EXAMPLE USING PAGING RATES

-Swap group size = 90 (storage isolated)

-Swap paging rate, SR = 400 pg/s

-Demand paging rate DR = 100 pg/s

Figure 9. PAGING RATE DESCRIPTION

In this section page/swap configuration alternatives are examined, assuming that significant paging loads are encountered. Figure 9 contains the description of the paging workload. Assume a total swap paging rate of 400 pages per second, and a demand paging rate of 100 pages per second, with a swap group size of 90 pages.

CONFIGURING FOR THE PAGING LOAD

The first step in configuring for swapping is to establish the number of paths required for good swap-in response times. A high level of parallelism is required for good swap-in response time. In order to obtain this, we divide the swap group size, SG, of 90 with the swap set size, SS of 30.

The minimum level of parallelism can be determined by dividing the swap group size, SG, with the swap sets size of 30. The minimum level of parallelism should be

SG / SS = 90 / 30 = 3.

The conclusion may be reached that from a concurrency viewpoint at least three independent paths are required for an all local paging environment.

The next step is to consider the number of 3380 actuators required for swap paging. The guideline says a potentially swapping actuator is capable of transferring up to SP=100 pages per second. Thus, the number of 3380 local page data sets required for swapping, NS, is four,

NS >= SR / SP = 400 / 100 = 4

i.e., at least four local paging actuators are required to accommodate the swap paging requirements.

The next step is to consider the number of 3380 actuators required for demand paging. The demand paging rate required is DR = 100 pages per second. A 3380 demand paging actuator can sustain a demand paging rate of DL = 25 pages per second according to the guidelines. Hence the number of local page data sets required for demand paging, ND, is four, because:

ND >= DR / DL = 100 / 25 = 4

The number of 3380 local demand paging actuators is additional to the number established by the swap paging requirements.

Thus, the total number of local paging actuators required is

NS + ND = 4 + 4 = 8

CONFIGURATION ALTERNATIVES

Now the configuration alternatives can be summarized. In the notation below "L" represents a local paging actuator, and "D" a non-paging actuator with data, such as TSO work packs. Paths are separated by commas.

1. All local paging on dedicated paths. The calculation showed the requirement for at least four demand paging and four swap paging actuators, for a total of eight local paging actuators on at least four paths. This exceeds the minimum parallelism requirement of three. Hence the configuration with dedicated paging paths is

   LL, LL, LL, LL

2. All local paging using non-dedicated paging paths. The calculation showed the requirement for at least four demand paging and four swap paging actuators, for a total of eight local paging actuators. We place them on eight non-dedicated paging paths. This exceeds the minimum parallelism requirement of three. Hence the configuration is

   LDDD, LDDD, LDDD, LDDD, LDDD, LDDD, LDDD, LDDD

   It is likely that this solution provides the better paging response times.

SUMMARY OF METHOD

A paging configuration design method was presented, assuming that the configuration is to be organized on the basis of paging load rather than space. The first step is to determine the paging load to be sustained. The next step is to determine the parallelism required, and the final step is to determine the number of paging actuators required to accommodate both swapping and paging.

Having determined the number of paging actuators and the required parallelism, the paging actuators can be configured on paths to meet the installation objectives regarding isolation of paging paths and/or isolation of data paths.

A CASE STUDY

In this chapter a case study of a 3090 installation is presented. The objective of the study is to describe some of the paging considerations in an environment with expanded storage. The installation had a 128Mb IBM 3090/400 configured with 128 MB expanded storage. There were a maximum of 430 TSO users on the system in the measurement period. An additional 40 batch initiators and 80 started tasks were also running. CPU utilization was about 93 percent.

THE PAGING ACTIVITY REPORT

Paging activity report

Rate of page movement to ES =417

Main Storage Paging Rates

	PAGE IN		PAGE OUT		Migration rate = 64			
	SWAP	NON SWAP	SWAP	NON SWAP	High UIC		Migr Age	
					Min	Avg	Min	Avg
NON-VIO	49	32	24	0.3	34	93	349	370

Figure 10. AN EXAMPLE OF PAGING ACTIVITY REPORT

An example of a RMF Monitor I Paging Activity report page I is given in Figure 10.

Observations.

- Expanded storage absorbs most of the potential paging i.e., 417 pages per second (out), and we may assume as a first approximation that a similar amount (in). (The assumption can be justified by saying that what goes out, tends to come in.) Contrast this with 49+32 total page-ins from auxiliary storage. The "hit ratio" on expanded storage is about 417/(49+32+417) = 84 percent.

- There is no shortage of central or expanded storage; since the UIC is greater than 30 seconds, and migration age is greater than 100 seconds. (Migration age is an indicator representing the life of pages on expanded storage in seconds before they are migrated. This number is an indicator of storage pressures on expanded storage).

- Swap-in from auxiliary storage (49 pages per second) is higher than swap-out to auxiliary storage (24). What? What comes in, must go out first. (The problem is that migrated swap-out pages are not fully counted in the swap-out category, only the migrated primary working set is counted).

- The page-in rate from auxiliary storage (32) appears relatively high in comparison with the swap-in rate (49). This question is worth addressing further. The immediate hypothesis is that there is no (or not enough) storage isolation, which inflates the page trims, that result in page-ins. The Installation Performance Specification (not shown here) confirms that there is NO storage isolation on the system.

- There is almost no page-out activity to auxiliary storage. (It goes to expanded storage first).

- The maximum paging activity can be roughly calculated. The expanded storage migration rate of 64 pages per second includes secondary working set swap-outs. Assume that it DOES NOT include primary working set swap-outs (which are included in the swap-out report). Thus an upper bound on (ASM) paging is

  49 + 32 + 24 + 0.3 + 64 = 169 pages per second

  If the system did not use swap data sets, the page data set report would provide an absolute count.

THE SWAP PLACEMENT ACTIVITY REPORT

Swap placement activity

Note: numbers are rounded for space reasons

	Tot Dir	Aux Swap	Log Dir	ES Aux Tot	ES Tot	ES Eff	Log Eff	Migr fr E	L/E Eff	%
Trm. I/O wait rate	16.1	.23	15.7	.21	.46	2.7	2.5	13.1	.19	15.6	97
Tot.	19.9	.23	19.5	.23	.56	3.0	2.8	16.6	.26	19.4	97

Average pages per swap-out = 44, per swap-in =87

Figure 11. AN EXAMPLE OF SWAP PLACEMENT ACTIVITY REPORT

An example of a RMF Monitor I Paging Activity report page 2, the Swap Placement Activity report is given in Figure 11.

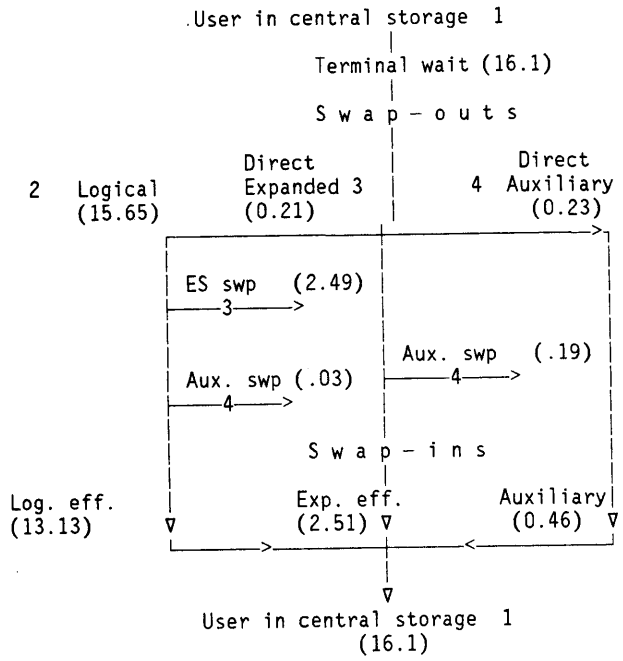


Figure 12. A representation of swap states

Figure 12 shows the various swap states in a state diagram. The user in central storage (1) enters terminal wait, therefore he is logically swapped (2), or direct swapped to expanded storage (3) or direct swapped to auxiliary storage (4). Both logical and expanded storage swaps can be converted into auxiliary storage swap-outs (4). Logical storage swap-out can also be converted into expanded storage swap-out (3). A user in any of the three swapped out states (2,3,4) ultimately reenters (swapped back in) into central storage (1).

The effectiveness of processor storage swap is indicated by the sum of the effectiveness of logical and expanded storage swaps.

Observations

- The terminal I/O wait rate is 16.1 per second, and auxiliary storage swap-in is only 0.46 per second, 2.8 percent of the total. Thus, auxiliary storage swap activity is not very heavy. The average swap-in size is the swap-in page rate of 49 divided by the aux. swap-in rate: of 0.56, which comes out to be 87 pages per second. (The average swap-in size is correctly reported on the report, but the average swap-out size is not).
- Non-swap page-ins represent an upper limit to swap trim. Thus, swap trim must be smaller than 32 pages per second. The overall swap rate is 20 transactions per second. Thus, if all paging came from swap trim, it would be (32 / 20 = 1.6), i.e., 1.6 page-ins per transaction.

RMF MONITOR I PAGE/SWAP DATA SET ACTIVITY REPORT

Interval = 15 (minutes)

Pag Spc Typ	Dev Num	Dev typ	% in Use	Pag XFR Tim	Rate I/O Req	Pag XFR Rate	Blk Fac tor	Allc Slts Thousands	Slts Used
LCL	1AA	3380	15	7	5.33	20.9	3.9	45	20
LCL	717	3380	28	17	4.57	16.5	3.6	45	20
(6 local data sets)									
Total						116			
					Serv Time				
SWP	627	3380	1.8	27	0.74				
(6 swap data sets)									

Figure 13. PAGE (SWAP) DATA SET USAGE REPORT FROM MONITOR I

Figure 13 shows extracts from the page and swap data set reports. The "I/O rate" and "blocking factors" columns do not appear on the report, the values in them were obtained by straightforward calculations[3].

Observations

- The use of the swap data sets is wasteful. Their utilization is very low, as expected in an expanded storage environment. The installation would gain by converting its swap data sets into local page data sets. Probably some of the twelve data sets could be used for other purposes.
- The two different local page data sets present very different measurements. Device 1AA has a reasonable average paging response time of about 30 ms, (3.9 times 7 milliseconds) device 717 has about double that

[3] Divide the reported page transfer count by the report interval duration to obtain the page transfer rate. Divide the reported page transfer count by the reported number of I/O requests to obtain the blocking factor.

amount. A further examination from the device and control unit reports is called for.

RMF MONITOR I DIRECT ACCESS DEVICE ACTIVITY REPORT

Dev CU Num	% in Use	Act Rate	Ave Resp Time	Ave IOSQ Time	Ave Pend Time	Ave Disc Time	Ave Conn
1AA	14	5.3	28	0	0.5	20.4	6.9
(1AA is typical paging device)							
00B		51.2	23	2	0.8	15.5	5.0
717	26	4.6	57	0	1.1	48.4	6.5
(717 is a paging device)							
707	94	22.3	162	117	2.1	35.9	7.7
(707 is a nonpaging device)							
024		190.2	50	25	1.7	18.2	4.4

Figure 14. DIRECT ACCESS DEVICE ACTIVITY REPORT FROM MONITOR I

Figure 14 shows extracts from the device activity report. The Logical Control Unit, (LCU) 024 consisting of two actual control units has high I/O activity rate (190 I/Os per second), which causes the paging device 717 to encounter a rather high disconnect time. (Disconnect time is the sum of latency, 8.3 ms plus RPS miss, the latter caused by path contention, plus seek time. Seek time on paging devices is normally very low, of the order of 3-4 ms). Observe that even device 1AA has high disconnect time, which is not easy to explain. One clue is obtained from the page data set usage report. On each local data set 45000 slots are allocated and about 20000 used. That is insufficient for the contiguous slot allocation algorithm to work effectively. As stated earlier, the number of available slots should be at least four times as large as the number of slots used. The allocation size should be increased to 64000 slots, and more local page data sets should be used. The excess latency and seek times would then be reduced.

CORRECTIVE ACTIONS FOR PAGING PROBLEMS

I want to emphasize that in a modern system environment paging problems can and should be avoided. When paging problems do occur, then they usually manifest themselves in slow TSO or DB/DC response times. Correcting these problems usually results in improved response time. Designing for response time is very important, since it results in better people productivity, which reduces people cost.

Paging problems can be reduced by providing additional hardware so as to reduce paging or make it faster or by tuning actions. These solutions are listed in the order they should be pursued.

The tuning solutions usually result in a better performing paging subsystem by reducing paging altogether, and as a next step, enhancing and increasing swapping and, simultaneously, reducing demand paging.

The solutions (below), have been presented before, here cognizance is taken of the presence of expanded storage.

1. REDUCE ALL KINDS OF PAGING.

   Problem: Heavy demand and swap paging, slow response times, slow page fault resolution.

   Solution: Increased processor storage (hardware).

   When processor storage is increased, paging rates (all kinds) are reduced. The effect is to decrease:

   - CPU cycles associated with paging.
   - The required number of paging devices.
   - The paging load carried by channels.

   When the processor storage used is at its limit, then (and only then) it is worthwhile considering other solutions.

2. CONTROL TSO DEMAND PAGING

   Problem: Heavy demand paging, slow TSO response times, slow page fault resolution.

   Solution: Introduce storage isolation for trivial TSO address spaces (tuning action).

   This can be done by using the PWSS parameter for trivial TSO transactions (i.e., for the trivial TSO performance group period). The effect is that demand page load is decreased, but swap load is increased. A further effect in an expanded storage environment is that the page-outs to auxiliary storage use the contiguous slot allocation algorithm, using the devices more effectively.

   Because of storage isolation the short term central storage occupancy of a TSO address space may increase, but the overall central storage occupancy (the frame-second product) decreases because of the shorter time spent in central storage waiting for page faults.

3. CONTROL DB/DC, JES, STARTED TASK DEMAND PAGING

   Problem: Heavy demand paging, slow DB/DC response times, hesitating printers, slow page fault resolution.

   Solution: Introduce storage isolation for these address spaces (tuning action).

   This can be done by using the storage isolation parameters PWSS and/or PGRT for the response time sensitive, usually nonswappable address spaces, such as IMS control region or message processing regions, CICS address spaces, or JES.

   The effect is that for the storage isolated nonswappable address spaces the demand page wait states are reduced, thus transaction response time improves. The processor storage occupancy of these address spaces becomes much more stable even though it may become larger. Batch and TSO swap load is increased.

4. REDUCE SWAP-IN DELAYS

   Problem: Slow TSO response times, large swap-in delays

   Solution: Provide maximum parallelism for swapping (tuning action).

   This means that the swap paging load is distributed to as many, non-dedicated paths as there are swapping actuators. In particular, use an all-local paging configuration, placing one or at most two such actuators per string-switched path complex. Result

is that more paths are used concurrently, and swap delays are reduced. As swapping takes place faster, the address space takes up central storage for a shorter period of time, and this in turn reduces demand paging. Central storage occupancy is measured in frame-seconds, and the shorter the period the address space spends in waiting for swap-in, (or page faults), the less real storage is occupied by that address space. As an example, an address space with 72 frames and a response time of 0.3 seconds occupies 72 x 0.3 = 21.6 frame-seconds. The transaction with a one second response time occupies more than three times as much. The smaller this figure is, the smaller impact it represents to the system.

Swap traffic may cause momentary delays for non-swap I/O on the non-dedicated paths, usually this is not significant however.

In an expanded storage environment (with swapping to expanded storage) generally do NOT use swap data sets, since only the primary working sets migrate to swap data sets.

5. CONTROL OVERALL DEMAND PAGING RATE

Problem: Slow TSO response times, large page-in delays

Solution: Change the IEAOPTXX PARMLIB member so that the SRM decides to swap-out long running TSO transactions and batch whenever page fault rate exceeds a threshold (RCCPTRT).

The result is reduced multiprogramming level, increased swapping and reduced paging. TSO response time improves.

6. CONTROL LOGICAL SWAPPING

Problem: Slow TSO response times, large page-in delays

Solution: Decrease the maximum system think time, LSCTMTE (tuning action) in an environment WITHOUT expanded storage.

When the "system think time" parameter is reduced to a small number, say, 1-5 seconds, from the MVS default of 30 seconds, only those users are logically swapped (in central storage only), who interact with the system very fast. As a result, the central storage occupancy of logically swapped users is reduced, the central storage available for frames containing unreferenced pages with steadily increasing UIC values is increased, and demand paging is reduced. Physical swapping will increase at the same time.

The rationale is that slow users do not deserve to occupy central storage when fast users are deprived. Do NOT use this tool in a system with expanded storage.

7. SPEED UP PAGING

Problem: Heavy demand and/or swap paging, slow response times, slow page fault resolution.

Solution: Enrich the paging subsystem (hardware). All the tuning actions recommended have been tried, the processor storage is at its maximum, and paging delays are still significant. Thus, more and faster paging devices should installed. If swapping is the problem, more 3380 actuators should be installed on more paths. If paging is the problem, replace 3380s with larger 3880-21s, (assuming no more processor storage can be obtained). (This problem probably occurs in non-3090 environments only).

CONCLUSIONS

The easiest way to solve paging problems is by obtaining more processor storage.

The most flexible paging configuration is an all local paging configuration.

For local page data sets the maximum local page data set size, 65400 should be allocated.

For good ASM swap response multiple swap paths are required. Thus, multiple local paging 3380 actuators distributed on non-dedicated paths should be used if possible.

In an expanded storage environment, ONLY ALL-LOCAL paging configurations should be used.

In an expanded storage environment, the use of nondedicated paging devices should be considered.

ACKNOWLEDGEMENTS

I want to thank my colleague, Mr. B. Pierce for reviewing this paper.

REFERENCES

1. T. Beretvas, "Page/Swap Configurations," G320-0053, IBM Corporation Marketing Publications, Department 824, 1133 Westchester Avenue, White Plains, N.Y. 10604

2. T. Beretvas, "Analysis of Processor Storage and Paging Configurations," CMG 1986 Conference Proceedings

3. A. J. Thadhani, "Interactive User Productivity," IBM System Journal, 20, No 4. 407-423 (1981).

4. P. J. Denning, "The Working Set Model for Program Behavior," CACM 11, 323-333 (May 1968)

IBM 3090 Processor Expanded Storage Experiments

John P. Pilch

Bell Communications Research, Inc.
Piscataway, New Jersey 08854

Dr. Bernard Domanski ***

The College of Staten Island / CUNY
Bell Communications Research, Inc.

ABSTRACT

The IBM 3090 processor series introduced a new architectural feature called Expanded storage. Prior studies at Bell Communications Research determined the performance of the 3090 model 200 relative to a 3084QX in Single Image mode, and of the 3090-200 relative to a 3090-400 in Single Image mode. These tests were conducted using MVS/SP 2.1 (MVS/XA) as the primary control program (both MVS 2.1.3 and 2.1.7 were used). The environments tested included a dedicated online IMS system (both IMS 1.3 and 2.1 were used) as well as a combined development and production scientific workload. This report documents our findings on the use of Expanded storage in a dedicated IMS environment, both as a paging device and as a Virtual Fetch device.

Significant performance improvements are seen with Virtual Fetch when Expanded storage is used. In the lightly loaded case, Schedule to First DL/I time is reduced 82% by using Expanded storage. In the heavily loaded case, Schedule to First DL/I time is reduced by 66% by using Expanded storage, and mean program execution time is reduced by 25.5%.

Expanded storage is an extremely fast paging device. However, it can be overcommitted. In one test designed only to produce demand paging, severe thrashing occurred as the ES rate approached 1800 pages per second.

1. Background

With the announcement of the model 200, IBM introduced the 3090 processor series, consisting of high capacity, single and multiple CPU general purpose computer systems. The IBM 3090 series introduced a new memory architecture option known as Expanded storage. With this option, two random access memories are present in the hardware configuration: the primary storage, called the Central storage (previously referred to by IBM as the Real storage), and a secondary storage, called the Expanded storage, used only for paging and swapping operations. The purpose of Expanded storage is to reduce the paging and swapping load to the channel attached Paging devices.

Two types of tests were conducted to determine the performance implications of using Expanded storage on a dedicated IMS environment. The first test is the use of IMS Virtual Fetch (potentially high VIO paging load) and the second test is the use of IMS Preload (potentially high demand paging load). Both of these tests are compared to the baseline case of IMS using Real Fetch.

The use of Expanded storage is explored here. The size of the Central storage (Real storage) is fixed at 64 Mbyte for the 3090-200 and this size was not an option.[1]

1.1 What is Expanded Storage?

Expanded storage is an architectural extension of the 3090 series processors. It is available as an optional feature in 64 Megabyte increments up to 256 Megabytes[2] (512 Mb on the 3090 model 400) in addition to the required 64 Megabytes (128 Mb on the 3090 model 400) of Central storage. Expanded storage is a high speed, reduced function memory designed specifically for paging and swapping operations to and from Central storage. It is called a *reduced function memory* because it *cannot* be used for program execution (active pages *must* be resident in Central storage), nor can it be used for I/O Operations (channel data paths are only connected to the Central storage). Transfers between Expanded storage and Central storage are performed in 4096 byte units (pages). Space in Expanded storage is managed with an approximation of the Least Recently Used (LRU) algorithm, the same as that used for Central storage. Therefore, though some paging occurs based on internal algorithms, almost no paging-related I/O occurs to the channel attached paging devices until Expanded storage is fully committed. Expanded storage is supported by MVS/SP 2.1.3 and is transparent to all application programs. Figure 1 depicts a high level view of the Expanded storage architecture.

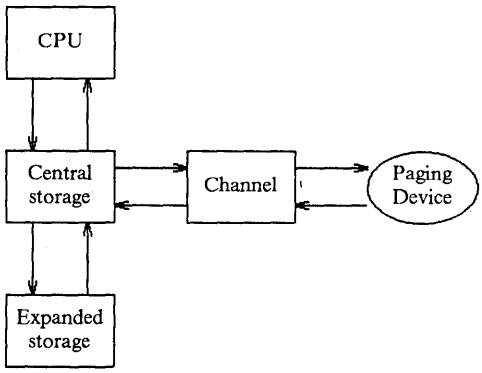


Figure 1

One might ask why Expanded storage was introduced, rather than just adding more Central storage. Even though the real storage cost per Megabyte declines, the complexity of *fully connecting* it to a system places limits on the price-performance improvement. Storage must be accessible to the I/O subsystem, as well as to all connected processors. When connected

1. The new 3090E series is available with either 64 or 128 Megabytes of Central storage.
2. Note that these numbers reflect the February 11, 1986 announcement by IBM; less was available at the time of testing.

*** This author is currently on a sabbatical leave from the College of Staten Island, a unit of the City University of New York. This work was conceived and performed while at Bell Communications Research, Inc., Piscataway, New Jersey.

to processors, the data flows through expensive cache buffers, so that instructions and data are passed at cycle speeds to the processors. As the workload continues to grow, so does the need for real storage. Increasingly high levels of multiprogramming make the storage reference patterns more random; thus, more circuitry must be devoted to cache to maintain acceptably high hit ratios. If continued improvements of fully connected storage approach a complexity barrier with respect to circuitry, then Expanded storage is a *partially connected* solution.

2. Terminology

To understand fully the performance effects of Expanded storage, some key definitions of measurement terms are in order:

- *ES Rate* (pages/second) - The rate of page movement from Central storage to Expanded storage.
- *Migration Rate* (pages/second) - The rate of page movement from Expanded storage to the Channel-connected paging devices.[3]
- *Migration Age* (time in seconds) - The amount of time a page resides in Expanded storage before migrating to the Channel attached paging device.
- *Criteria Age* (time in seconds) - A user specified parameter used to determine if a page being removed from Central storage is to be sent directly to the Channel attached paging devices rather than being sent to Expanded storage.[4]
- *Page In* - The movement of a page from the channel attached paging devices to the Central storage.
- *Page Out* - The movement of a page from the Central storage to the channel attached paging devices.

3. Description of IMS Workload - Small Memory Load

There are two primary variables in this study, the use of Virtual Memory by the online IMS system and the resulting load on the paging subsystem. The use of Virtual Memory by MVS, VTAM, etc., remained constant throughout the study and is not described in this report.

The online application used in these tests was run under the control of MVS/SP 2.1.3 and IMS 1.3. The application consisted of 150 transactions and programs run at volumes high enough to have the CPU utilization between 75% and 90%. The program size ranged from 19 Kbytes to 455.6 Kbytes. Since all programs were not run with equal frequency, the average size of a program fetched for execution was 54 Kbytes. The total size of the 150 programs executed during the tests was 12,212 Kbytes. For the application programs to execute correctly, 1,496 Kbytes of subroutines must be preloaded into each processing region. Accounting for the various IMS modules that must be present in the Dependent Regions, the maximum size of a dependent region is 2,028 Kbytes for the Real Fetch and Virtual Fetch tests.

The total Virtual storage requirements for the IMS system are:

- Control Region - 2,124 Kbytes
- DL/I Address Space - 7,440 Kbytes
- DBRC - 500 Kbytes
- Dependent Regions - 28 regions at 2,028 Kbytes each = 56,784 Kbytes.

This IMS system, run on an IBM 3090-200 with 64 Mbyte Central storage and no Expanded storage, produced almost no paging activity (less than 0.1 pages/sec.) over all volumes tested.

3.1 Real Fetch

The baseline set of tests was the IMS application described above, run on 64 Mbytes of Central storage and no Expanded storage. Five copies of the program load library were alternated among the 28 message processing regions to minimize contention. Schedule to First DL/I time averaged 77.76 milliseconds and there was no paging to the channel attached paging devices.

3.2 Virtual Fetch Description

Virtual Fetch is a feature of the MVS contents supervision function. IMS releases 1.2 and above have been written to exploit this MVS feature at the option of the particular installation. Virtual Fetch was written to address the problems associated with MVS fetch and IMS Preload. Virtual Fetch executes in an independent address space, thereby providing Virtual Fetch services to all other requesting address spaces. Load modules that are specified by the installation are loaded and maintained by Virtual Fetch in VIO datasets created by Virtual Fetch initialization. Modules managed by Virtual Fetch may be invoked by *any address space* with Virtual Fetch requests. When an application requests Virtual Fetch services, Virtual Fetch pages the specified module into the requester's address space and passes control to it. Thus, by using both VIO and Cross Memory Services in MVS, Virtual Fetch provides yet another mechanism for managing IMS online application programs.

IMS Dependent Regions use Virtual Fetch services when the installation provides an IMS Parmlib member with the names of the particular programs to be processed by Virtual Fetch. Each dependent region can have a different list of Virtual Fetch programs. In our tests, each of the 28 dependent regions were given a list of all 150 programs that make up the application. Therefore, Virtual Fetch managed all 150 programs totaling 12,212 Kbytes in the VIO dataset.

3.3 Virtual Fetch Analysis

Prior investigations conducted at Bell Communications Research have assessed the relative performance of the different program load methods. The IMS trace facility (DCMON) provides a measurement called Schedule to First DL/I Call time.[5] This measurement reports the time between the dependent region being told which application program to execute and the application program making its first DL/I Call (usually a call to receive the message text). One function that is performed in this interval is locating and placing into memory the required application program. We use Schedule to First DL/I as an indirect measure of the efficiency of the various program management techniques. Measurements on previous processors show that Real Fetch is the slowest method, Virtual Fetch takes roughly half the time of Real Fetch, and Preload takes roughly half the time of Virtual Fetch.

Table 2 illustrates the results of our Virtual Fetch experiments with Expanded storage. The value for Schedule to First DL/I Call using Virtual Fetch on the 3090/200 *without* any Expanded storage was 47.84 milliseconds; this is 61.5% of the corresponding value (77.76 msec) found for Schedule to First DL/I Call using Real Fetch on the 3090-200. Note, though, that the paging rate had increased from 0 to 337 pages per second. When 64 Mb of Expanded storage was added, the value of Schedule to First DL/I was reduced to 8.72 msec, a reduction of 89% over Real Fetch, and a reduction of 82% over Virtual Fetch without Expanded storage. Note that because Expanded storage was *not full*, almost no I/O's were performed to the paging subsystem. Also note the negligible effect on both Schedule to First DL/I time and on the paging rate when Expanded storage was increased from 64 Mb to 128 Mb. This data is presented graphically in Figure 3.

Effects of Virtual Fetch on Program Load Time

Environment	Expanded Storage (size)	Schedule to First DL/I (msec)	Paging Rate IN + OUT (per second)
Real Fetch	0 Mb	77.76	0.11
Virtual Fetch	0 Mb	47.84	337.08
Virtual Fetch	64 Mb	8.72	0.12
Virtual Fetch	128 Mb	8.97	0.18

Table 2

3. Since an address in Expanded storage cannot be used as the source or destination of an I/O operation, an intermediate move of the data is made to Central storage before the physical I/O operation is performed to the channel attached device.
4. See section "Controlling Expanded storage" for the use of this parameter.
5. See: S. Hoyt, "IMS/VS/MVS Performance and Tuning Guide, IBM Corp., G320-6004, April, 1986, for a complete definition.

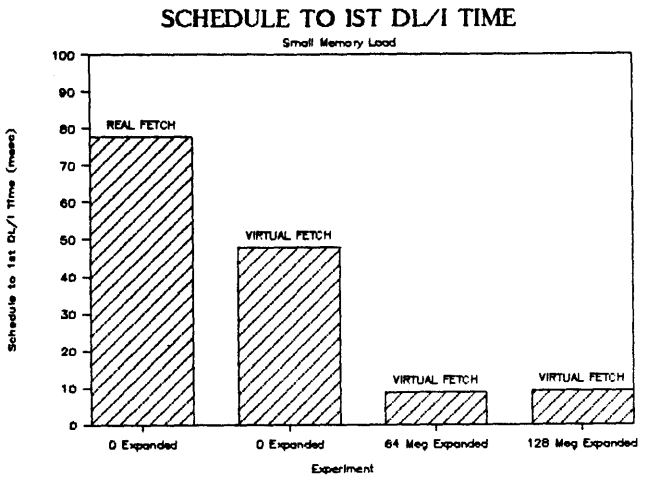


Figure 3

3.4 Summary - Small Memory Load

Virtual Fetch provides a performance improvement over Real Fetch. However, a considerable load is placed on the paging subsystem.

In this section, the functionality of Expanded storage in isolation was tested. The memory commitment and the paging subsystem were not excessively stressed. In the next section, we determine if the performance improvements gained through Expanded storage are maintained when the memory is overcommitted, and the paging subsystem is stressed.

4. Description of IMS Workload - Large Memory Load

For the IMS Preload experiments, we would have liked to keep the workload constant. However, if all 150 programs were to be preloaded, this would have required an MVS private area greater than 12 Mbytes. Instead, we preloaded *60* programs and used Real Fetch on the remaining *90* programs, and maintained the same hourly volumes as the previous experiments. As before, the program size ranged from 19 Kbytes to 455.6 Kbytes. The total virtual memory required for each dependent region is 6964 Kbytes, allocated as follows: 4884.8 Kbytes for preloaded programs, 1496 Kbytes for preloaded subroutines, 455.6 Kbytes for fetched modules, and the remainder for the region controller. Using 28 dependent regions, the total Virtual memory requirement for the message regions is 194,992 Kbytes (28 * 6964) or 190.422 Mbytes. To calculate the total Virtual memory requirement, we add *2124* Kbytes for the IMS Control Region, *7440* Kbytes for the DL/I Address Space, and *500* Kbytes for the DBRC Address Space. This yields a total Virtual memory requirement of *205,056* Kbytes, or *200.25* Mbytes. To put this in perspective, our moderately loaded systems' average Virtual memory requirement is 140 Mbytes.

4.1 IMS Preload Description

IMS Preload operates in a different fashion than either Real Fetch or Virtual Fetch. Rather than loading a module at each program scheduling, IMS Preload loads installation-specified modules into the dependent region private area at the time of initialization of the dependent region. Once a module has been preloaded, its starting address is saved in a table; later requests for the scheduling of an IMS Preload-managed module results in a table lookup and a direct branch to the module's virtual storage address.

Unless a program is reasonably active throughout the life of the dependent region, its pages will age and will most likely be replaced or paged-out, depending on whether they have been updated. When IMS branches to one of these "aged" modules, a page-fault occurs, generating a request to page in the required page frame; this causes a delay in processing owing to page fault resolution. In addition, each reference to a page of the program that has been paged-out to "auxiliary" storage causes a page fault. This is known as *serial page faulting*, and usually causes an increase in both Schedule to First DL/I time and program elapsed time. Finally, significant stress is placed on the paging subsystem, eventually affecting the entire MVS system.

4.2 Analysis of Program Load Time

The first set of experiments was designed to determine the amount of relief Expanded storage provides to the channel attached paging subsystem. Here, identical workloads were executed with and without Expanded storage to compare directly the effects on the paging subsystem. To increase the load on the paging subsystem, 60 programs were still preloaded into each processing region, and the remaining 90 programs were placed under the control of Virtual Fetch. This results in the same virtual memory commitment (200.25 Mbytes) as described above. The paging subsystem in these experiments consisted of two 3880-model 21 paging controllers (each with 32 Mbytes of cache), and four 3350s containing local page datasets. The baseline case was run with no Expanded storage. The second case was run with 64 Mbytes of Expanded storage.

Without Expanded storage, the total paging load was 370 pages per second, 222 pages/second for VIO, 73 pages/second for Page In, and 75 Pages/second for Page Out. Schedule to First DL/I Time for Preloaded programs averaged 89.04 milliseconds, while Schedule to First DL/I Time for the programs under the control of Virtual Fetch was 63.36 milliseconds. This is counter to previous findings.

Demand paging and VIO were not segregated in the configuration. Both types of paging activity competed for device service and space in the cache controller. In our case, a Virtual Fetched module is referenced 18.67 times more frequently than a preloaded module[6]. We conclude the Schedule to First DL/I time is shorter for Virtual Fetch because of a higher read hit ratio in the cached paging controller.

With Expanded storage, the total paging load was 25 pages per second, 18 pages/second for VIO, 3 pages/second for Page In, and 4 pages/second for Migrate Out. Schedule to First DL/I Time for the preloaded programs was 9.11 milliseconds and for Virtual Fetch it was 21.70 milliseconds. By relieving the congestion on the channel attached paging subsystem, the relative performance of preload versus Virtual Fetch has been restored (i.e. preload is 58% of Virtual Fetch). Though both sets of transactions benefit greatly from the addition of Expanded storage, the Preloaded transactions experience a *90%* reduction in Schedule to First DL/I time, while the Virtual Fetch transactions see a *66%* reduction. These results are summarized in table 4 and figure 5 below.

Analysis of elapsed time without Expanded storage showed that the mean program execution time was 795.81 milliseconds; with Expanded storage, this was 592.12 milliseconds. This is a decrease of 25.5%. At the 90th percentile level, the program execution time 2.05 seconds without Expanded storage and 1.5 seconds with Expanded storage. This is a decrease of 26.8% indicating that all programs receive the same benefit from Expanded storage.

An additional experiment with real memory overcommitted had 60 programs preloaded into each region, and the remaining 90 programs under the control of Real Fetch. This experiment was designed to be compared with the Real Fetch and Virtual Fetch experiments of the previous section. This experiment was conducted on a 3090-200 with 64 Mbytes of Central storage and 64 Mbytes of Expanded storage. The channel attached paging subsystem was a 3880-21 with 32 Mbytes of cache and four 3350s with local paging datasets. Preloaded programs averaged 9.87 msec Schedule to First DL/I time while the program using Real Fetch averaged 84.77 msecs. Although the Central storage was overcommitted, the use of Expanded storage minimized the activity to the channel attached paging subsystem. The total load on the channel attached paging subsystem was 28 pages per second, 12 pages/second for page in and 16 pages/second for migrate out. When compared to the results of the previous section, Schedule to First DL/I Time increased 7 milliseconds (9%) for the programs under the control of Real Fetch. Preloaded programs in the stressed case were slightly slower (1.15 milliseconds) than the Virtual Fetched programs in the unstressed case with Expanded storage. Because of time constraints, this experiment was not repeated without Expanded storage.

Effects of Virtual Fetch vs. IMS Preload			
Environment	Expanded Storage	Schedule to First DL/I (msec) Preload	Fetch
Preload/Virtual Fetch	.0 Mb	89.04	63.36
Preload/Virtual Fetch	64 Mb	9.11	21.70
Preload/Real Fetch	64 Mb	9.87	84.77

Table 4

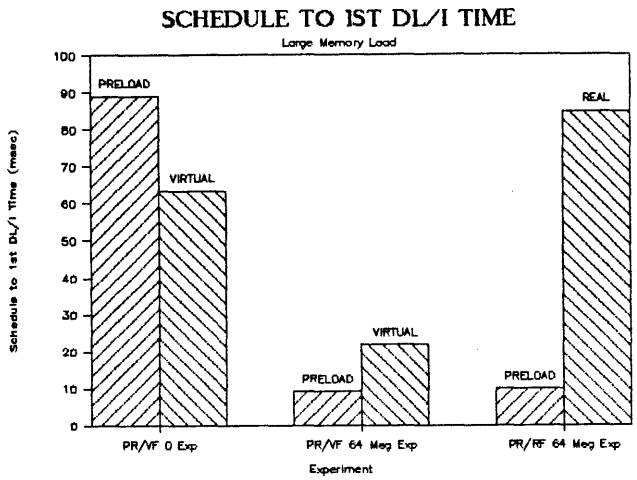


Figure 5

4.3 Analysis of Paging Subsystem

Expanded storage was designed to reduce the load on the channel attached paging subsystem. From the experiments conducted, the most direct comparisons can be made from the Preload-Virtual Fetch runs, with and without Expanded storage.

Without Expanded storage the total paging rate was 370 pages per second. With Expanded storage, the rate dropped to 21 pages per second. This is a reduction of 94.3%. VIO page in was reduced by 91.9%, demand page in was reduced by 95.9%, and demand page out was reduced by 100% (i.e page out was zero). See table 6.

Along with this reduction in paging rate, there are similar savings at the device level. The device activity rate (totaled for all devices in the paging subsystem) dropped from 92 operations per second to 22 operations per second when Expanded storage was added (72% reduction). Device response time dropped from 31 milliseconds to 3 milliseconds (93% reduction). Page transfer time dropped from 8 milliseconds to 1 milliseconds (87.5% reduction).

Although there were significant changes to the channel attached device activity, the activity in and out of Central storage remained the same (as one would expect). One measurement (activity from Expanded to Central storage) is not recorded. Therefore our conclusion can be inferred, not proved, from the data.

Without Expanded storage the demand page in rate was 73 pages per second and the demand page out rate was 75 pages per second, or nearly the same. With Expanded storage, the page in rate was 3 pages per second and the migrate out was 4 pages per second, again nearly the same. The 75 page out operations per second without Expanded storage went to zero with Expanded storage. However their movement out of Central storage is recorded in the ES Rate (movement from Central to Expanded). Accounting for the 4 pages per second that eventually migrate out to the DASD, we have a Central storage page out rate of 74 pages per second, nearly identical to the page out rate in the case without Expanded storage. Since in the case without Expanded storage, the in rate equaled the out rate, we assume these 74 pages per second are referenced within the migration age and are moved back to Central storage (the equivalent of an ES IN which does not exist as a recorded measurement).

Paging Subsystem Analysis				
		PRELOAD / REAL FETCH 64Mb Central / 64Mb Expanded	PRELOAD / VIRTUAL FETCH 64Mb Central / 0Mb Expanded	PRELOAD / VIRTUAL FETCH 64Mb Central / 64Mb Expanded
VIO Rate	(per sec.)	0	222	18
Page In Rate	(per sec.)	12.47	73	75
Page Out Rate	(per sec.)	0	75	0
ES Rate	(per sec.)	75	0	78
MIG. Rate	(per sec.)	16	0	4
High UIC	(seconds)	34	46	50
Dev. Activity Rate	(per sec.)	14	92	22
Response Time	(msec.)	18	31	3
Page Transfer Time	(msec.)	10	8	1

Table 6

5. Control of Expanded storage

The primary objective with respect to using Expanded storage efficiently is to send to Expanded storage those pages that will be referenced *before* migrating to channel-attached paging devices. Several variables can be set in the IEAOPTxx member of SYS1.PARMLIB to help control the use of Expanded storage.[7] These parameters are summarized below.

6. In total, there are 1770 copies of the programs that were executed, 90 programs in the Virtual Fetch library and 1680 (28*60) programs in the dependent regions. For an arbitrary program to execute, it is 18.67 times more likely to reference a copy in the Virtual Fetch VIO dataset than a preloaded program in one of the 28 separate address spaces.

7. IBM, "MVS/Extended Architecture System Programming Library - Initialization and Tuning", GC28-1149-2, 3rd Edition, July, 1985

Criteria Parameters for Expanded Storage	
ESCTPOC	Criteria age (secs) of changed page-out pages
ESCTPOU	Criteria age (secs) of unchanged page-out pages
ESCTSTC	Criteria age (secs) of changed stolen pages
ESCTSTU	Criteria age (secs) of unchanged stolen pages
ESCTSWTC	Criteria age (secs) of changed swap-out pages
ESCTSWTU	Criteria age (secs) of unchanged swap-out pages
ESCTSWWS	Criteria age (secs) of working set pages for swap-out
ESCTVF	Criteria age (secs) of Virtual Fetch pages

Table 7

To determine when page frames are moved to Expanded storage, consider the following:[8]

- *For SWAP pages:*

  IF (HI UIC + migration age) > criteria age THEN
  page movement to Expanded storage

- *For Terminal Wait pages:*

  IF (HI UIC + migration age - user Think time) >
  criteria age THEN page movement to Expanded storage

- *For stolen, Virtual Fetch, or pageout pages:*

  IF migration age > criteria age
  THEN page movement to Expanded storage

Again, the purpose of these calculations is to send only those pages to Expanded storage that are likely to be referenced again before being migrated to auxiliary storage. The criteria age values are nearly constant; they can be changed only by changing the variables specified above in PARMLIB. The HI UIC and the migration age are dynamic, and they change constantly to reflect the current workload on the system.

6. Effects of Expanded storage

To evaluate the consequences of Expanded storage on paging, we need a model of process behavior. Let us define the following parameters that characterize Central storage:

t	time to access memory in primary storage
T	time to access secondary storage
s	fraction of the program (+ data) resident in primary storage
$p(s)$	number of page faults per memory reference as a function of fraction of program that is resident

Table 8

The effect of demand fetching is to increase memory access time from t to $t + p(s)\ T$. Figure 9 characterizes the behavior of $p(s)$.

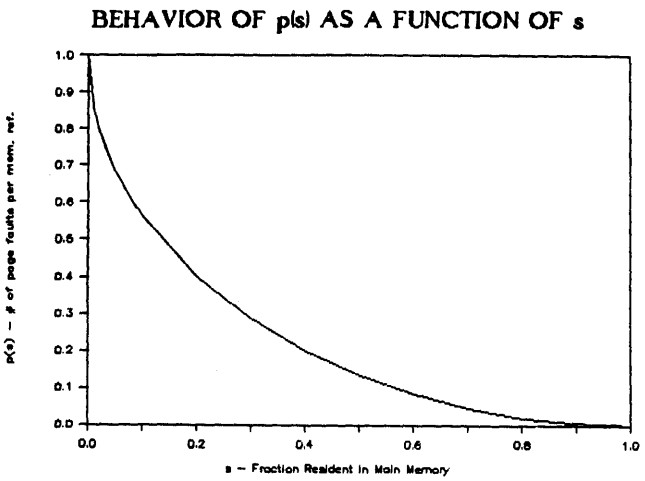


Figure 9

Coffman & Varian[9] performed many experiments on an IBM 360/50 to study empirically the effects of paging on program behavior. Their study led to several conclusions:

1. The greater the available memory, the greater the percentage improvement in page faulting.
2. Increases in memory are more beneficial than increases in page size when control frequently resides in a set of instructions or data that is small compared with the page size.
3. References to pages containing data induce substantially more paging than do references to pages containing instructions.
4. The *tail* of $p(s)$ can be approximated by

$$p(s) = ae^{-bs}$$

where

$$0 < a < 1 < b \quad \text{and} \quad a\,e^{-b}\,T << t$$

Denning[10] defines the *working set*, W, as the "minimum collection of pages that must be loaded in main memory for a process to operate efficiently without unnecessary page faults". For purposes of analysis, we must quantify the working set.. We define the working set W as that fraction of the program resident in the main memory such that the time between page faults equals the page fault service time. As an equation, we have:

$$p(W) = \frac{t}{T}$$

From the approximation of $p(s)$, we derive

8. In these formulas HI UIC is the High Unreferenced Interval Count as calculated by the system, Migration Age is the migration age calculated by the system, and Criteria Age is the value specified in the IEAOPTxx member in SYS1.PARMLIB.

9. Coffman, Varian, "Further Experimental Data on the Behavior of Programs in a Paging Environment", Communications of the ACM, Vol. 11, No. 7, July, 1968, pps. 471-474

10. Denning, P. J., "The Working Set Model for Program Behavior", Communications of the ACM, Vol. 11, No. 5, May, 1968, pps. 323-333

$$W = \frac{1}{b} \ln\left(\frac{aT}{t}\right)$$

As no guidelines appear for values of a and b, we borrow an example from Brinch Hansen[11] where $a = 0.1$, $b = 10$, $t = 1\ \mu sec$:

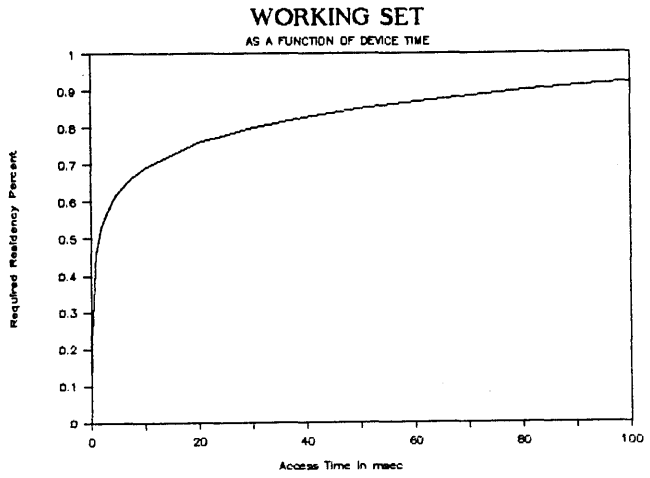


Figure 10

Figure 10 depicts the relationship between the speed of a paging device and the corresponding working set percentages (the minimum amount of program that must be present in primary storage to prevent thrashing expressed as a percent of total program). Thus, for the following set of possible paging devices with corresponding access times[12] we derive corresponding working set percentages:

Paging Devices & Access Times		
DEVICE	ACCESS TIME *(T)*	*W*
IBM 3380 disk	20-30 msec	76% - 80%
IBM 3880-11 (or 21)	8 to 10 msec	67% - 69%
STC 4305 (Solid State)	3 - 5 msec	58% - 62%
Expanded Storage	0.015 - 0.115 msec	4% - 24%

Table 11

Thus, the above example suggests that a faster secondary store (e.g., disk, drum, Expanded storage) allows a *smaller working set* per program at a given level of performance. Denning points out that a smaller working set allows a *higher multiprogramming level* with a given size memory. Therefore, Expanded storage appears to increase the potential concurrency level for a system of a fixed memory size.

The above analysis should only be used as an intuitive guide to the performance of Expanded storage, as

- No recent approximation for *p(s)* exists.
- No IBM published figures exist for memory access time *t* on a 3090.
- This analysis assumes all programs have identical values for *p(s)* and that *p(s)* is time invariant.
- This analysis does not account for Central storage requirements for page migration.

Although this section provides motivation for the use of Expanded storage, it does not answer the important question of which is better to acquire; Central storage or Expanded storage. When the 3090 series was first announced, this question was moot, Central storage was offered in one size only. The 3090 E series now provides options for Central storage. Experiments are being planned for the summer of 1987 to compare Central to Expanded directly.

We conclude this section with a quote from Denning: *"There should be enough core to contain a program's working set. There is no substitute for core."*

7. Conclusions

After careful examination, we find that Expanded storage does reduce the excessive demands on the paging subsystem. One can consider Expanded storage to be an extremely fast paging device; that is, page faulting will still occur when references are made to Central storage that cannot be resolved. However, if the referenced page frame resides in Expanded storage, the page reference is resolved at a speed that is much greater that of the more traditional paging subsystem. The best page response time for a channel-attached solid state device is reportedly less than 10 milliseconds; Expanded storage appears to be resolving page faults in the microsecond time frame. But because the next level of the paging hierarchy responds in a few *milliseconds*, there can be a sharp break in page fault resolution time in a system with fully occupied Expanded storage.

Expanded storage *can* be overcommitted. In one case where the ES Rate approached 1800 pages per second, the CPU was fully committed to serving page fault resolution (i.e. 100% CPU utilization with no throughput). The user of Expanded storage is encouraged to tread carefully in these new waters.

8. Acknowledgements

We would like to take this opportunity to thank all those IBM personnel who helped coordinate this effort; their support was essential for the completion of the project. Thanks also go to those Bell Communications Research personnel who provided both technical and administrative support. Any questions or comments on the material presented here can be addressed to one of the authors.

11. Brinch Hansen, P., "Operating System Principles", Prentice-Hall, 1973

12. The access time range for Expanded storage was estimated by the Candle Corporation in the Candle Computer Report Vol. 7, No. 13, July, 1985

IS A PICTURE ALWAYS WORTH A THOUSAND WORDS

Douglas R. MacKinnon, Ph.D.
Candle Corporation

ABSTRACT

Increasingly, graphics are being used in management level presentations and reporting systems. It is important that graphics be used for an express purpose and not in a casual manner to assure that they honestly reflect the data and make a clear point. This paper discusses a variety of common graphical errors and offers suggestions to clarify the use of pictures to represent data.

INTRODUCTION

As performance analysts or capacity planners, our jobs center around analyzing data and conveying our findings to others. The techniques we use are usually verbal descriptions of data, tables, and pictures. Each of these techniques is useful where appropriate. Verbal descriptions can explain, interpret and express conclusions. Tables present exact information in a concise manner to allow flexible comparisons. Pictures are powerful ways to communicate comparisons and relationships. Increasingly in data processing and business in general, graphics are being used in management level presentations and reporting systems.

There are a variety of reasons why graphics are becoming more commonplace. The first reason is that they are so much easier to produce today than in the past. Prior to the days of PC's and mainframe graphics packages, the analyst had to rely on his own skill at mechanical drawing or the services of an artist or draftsman. In either instance, the process was time consuming and labor intensive. As a result, graphs were produced only after careful study of the data to highlight some particular points of value. Today, in a matter of moments, any analyst can generate reams of graphical reports and select those that seem most promising.

Another reason for the popularity of graphs appears to be based upon the belief that graphics makes a presentation more professional looking. This idea may have originated when graphics were difficult to produce, but with their proliferation as a result of inexpensive PC software, today graphics are considered no more professional than tables.

Finally, there is the belief that the use of graphics keeps the audience awake and simplifies the presentation because graphs are easier to understand than tables. Actually, such a general statement is not supported by research. In general, the literature is contradictory, with some studies supporting graphics and others supporting tables. A recent study by Dickson, DeSanctis and McBride (1986) provides some insight into the contradictory nature of some of the research. Essentially, they conducted three separate experiments of differing levels of complexity with business school subjects. Their general finding was that graphics are superior to tables in supporting decision making in only a few instances: those involving large amounts of data over time; and where the audience is presented with large amounts of data and the goal is to have them recall a few specific facts immediately after the presentation. Essentially, their conclusion was that graphics are not generically superior to tables, and the correct technique depends on the task at hand. Conflicts in the results of earlier research probably were due to the differences in the tasks being performed by the subjects.

These conclusions have a significant implication for the analyst: there should be an express purpose behind producing a chart of graph. Graphics should not be the default method of conveying the meaning of data, but instead should be used only after text and tables have been expressly ruled out as inappropriate.

THE CHOICE OF TEXT AND TABLES

In order to determine if graphics is the preferred mode of data presentation, there are some simple guidelines. Where the purpose is to explain, interpret or explain the data, text is the vehicle of choice. When the intent is to compare only two numbers, text is also an appropriate medium. For example:

> The number of concurrent TSO sessions increased from 200 in November to 225 in December.

When several numbers are being conveyed, a simple table becomes more appropriate to facilitate comparisons within the data. For example,

> The accuracy of the forecasts for CPU use by our major divisions in the past calendar year were as follows:

Manufacturing	98%
Marketing	89%
Systems	55%

In this situation, there is little to be gained by using a chart as opposed to this simple table. In addition, when the accuracy of the data is important or when many localized comparisons are being made, tables are the appropriate technique. For example, if the intent is to present the total billing of data center services for batch and each online application within each department for the company, a super table is more preferable to hundreds of trivial bar charts.

In addition, there is a class of hybrid tables such as the stem and leaf table below (Tukey, 1977).

Age Distribution of Attendees at CMG

```
0 I
1 I
2 I 3446778999
3 I 01112446667788899
4 I 001122333445556666777778999
5 I 0245578
6 I
```

This table combines a table's ability to preserve all the granularity of the data with the ability of a frequency chart to display the shape of a distribution. The numbers on the left represent units of tens. The numbers on the right represent individual observations. In this instance, there were no values less than 10 or from 10 to 19. The smallest observation was 23, followed by two 24's, a 26, two 27's, etc. Most of the values were beteeen 40 and 50, and none exceeded 58.

PURPOSE OF GRAPHICS

There are two situations where pictures are most applicable: in expressing relationships between variables, and in emphasizing the key points the analyst wants the audience to recall from the presentation. Whatever the original motivation of the analyst to consider graphics, the actual utility of graphics should be to reduce the complexity of the data by guiding the audience to make the appropriate comparisons and recognize the trends in the data. The concern in developing graphics is not to be guilty of making the interpretation of the situation more complicated. There are two essential principles to follow in producing viable graphics: First, honestly portray the data; and second, make your point as clearly as possible. Discussing these two principles will provide the focus of the rest of this paper.

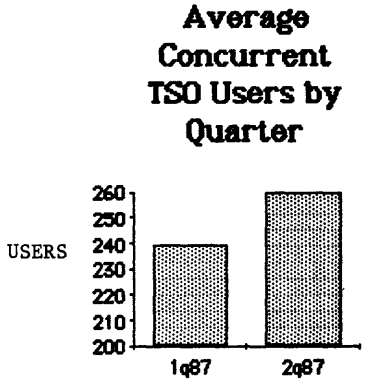
HONEST PORTRAYAL OF THE DATA

Honestly portraying the data in pictures requires avoiding those presentation errors that can lead to visual deceptions. Essentially, deception occurs when the graph or chart misleads the eye into interpreting that the relationship displayed is more or less extreme or otherwise different than is actually the case. As we produce and evaluate our graphics, there are five basic classes of these common errors: scaling, perspective and perception, standardization, data integrity, and data ordering.

SCALING

Scaling of the vertical and horizontal axis is probably the most common source of error that occurs in graphics. The types of errors that occur are essentially those of an inappropriate choice of range of values, mixing different scales on the same chart, and changing the base of the scale. In example 1a, the average number of TSO users per quarter is displayed. It would appear that in just the past six months that there has been a 50 percent growth in concurrent number of TSO sessions. However, such is not the case, because inspection of the scale of the vertical axis shows that the origin is set at 200 concurrent sessions.

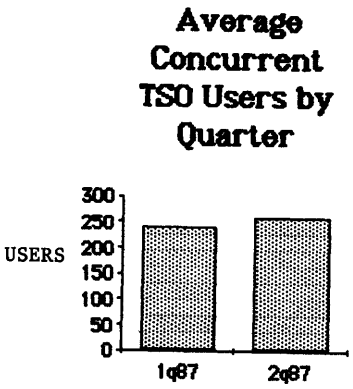
Example 1a



In example 1b, which is drawn to scale, it is clear that there has been growth in the number of sessions, but that growth is on the order of less than 10 percent. Example 1a basically exaggerated the growth in workload five fold over what would be supported by the data. For the analyst to suggest that the labeling was clear enough to have avoided an erroneous conclusion, misses the basic point: the picture must clearly represent the extent of the comparison, while the labeling is simply supporting information.

In addition, in presentations to large audiences who may not have copies of the presentation, notations and scales are often not legible; thus, the audience is left to make its interpretations and conclusions largely without the benefit of copies of the graphics. Particularly in these cases, the pictures must be able to stand alone in correctly illustrating the extent of the comparison or the trend of the data.

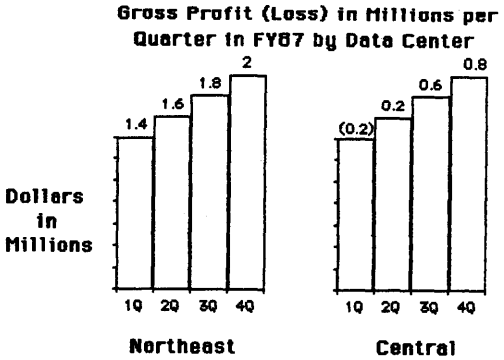
Example 1b



As a general rule, all charts which seek to illustrate the relative differences in values should explicitly show the zero point without a broken scale. One possible exception where a broken scale would not compromise the integrity of the chart is where the purpose of the chart is to highlight similarities and differences in the slopes of a relationship between two variables.

Not all changes in scaling are as easy to catch. In example 2, we see that the gross profit for two data centers are being compared. The initial interpretation is that each center has experienced healthy profits over the last four quarters. Implicit in this chart is that the scales are the same for each data center, which is not true. Hidden, for the audience to discern in the short time a slide is displayed, is the fact that the vertical scales are different in each comparison and, in fact, in the second data center, the gross profit was negative in the first quarter.

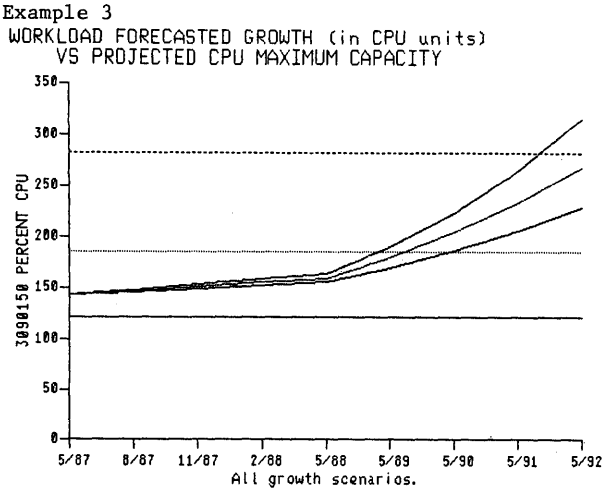
Example 2



Problems in scaling are not limited to the vertical axis alone. The following illustration, in example 3, was generated using a major capacity planning package and appears to suggest that the current configuration will meet the company's needs over the next two years

but will then quickly run out of capacity. However, the scale shifts from quarterly data to annual data halfway along the scale. Obviously, the data center configuration is not really in imminent need of more hardware, and, in fact, will experience a degradation in performance much more slowly than is implied by this graph. Some analysts might suggest that such a chart would be appropriate with adequate labeling or by slightly separating the graphs. Yet, particularly if such a graph is intended for a foil or slide presentation, such annotations would be largely unseen by the audience and still mislead the viewer about the extent of the basic trend of the data.

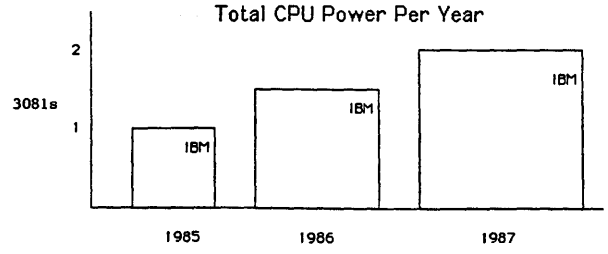
Example 3



PERSPECTIVE AND PERCEPTION

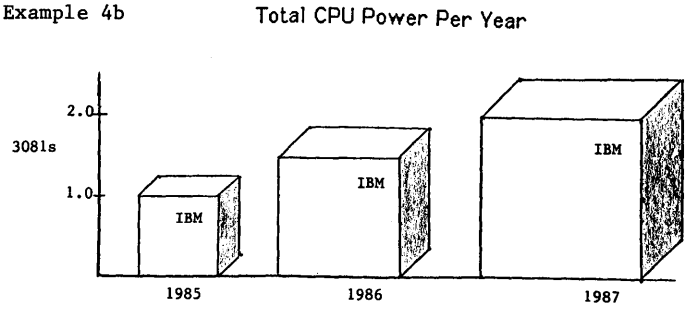
Two and three dimensional graphs are becoming more commonplace because of their ready presence in graphical and drawing packages. In general, these techniques are used to make the presentation "more interesting." Yet, the use of such techniques probably lead to more problems than they overcome. The general difficulty occurs when one dimensional data is presented using figures that change in two or even three dimensions at the same time. In addition, their use in these situations leads to a considerable increase in clutter with no increase in information being conveyed. In the next example, the analyst hoped to stimulate the audience's interest by presenting some "snazzy pictures" to keep the audience awake. Rather than use a table or a simple bar chart, he decided to use pictures of CPU's to portray the growth in the data center's compute power.

Example 4a



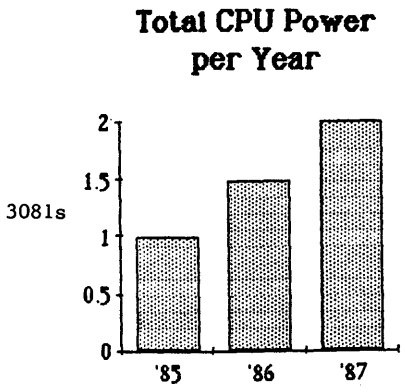
In example 4a, the height of the CPU corresponds to the total compute power at a given point in time. Unfortunately, the eye tends to interpret and compare differences in total area from one time period to another. In this simple comparison, where the total compute power doubled, the graph portrays two pictures that differ in area by a factor of four. In this instance, the audience may understand and be able to recall that the growth in compute power was considerable. It is unlikely that they will correctly recall the extent of that growth and will likely overestimate the size of that growth.

Example 4b



With the advent of three dimensional graphics packages, this type of error can be even more compounded with a three dimensional representation of the CPU growth as in example 4b. In this instance, although the size of the data center doubled, the volume of the CPU graphic increased by eight fold. In neither of these two cases has the analyst been true to the data. Yes, pretty pictures have been created, but the audience has been mislead and nothing was added beyond the much simpler presentation of example 4c. Most likely, the analyst would have been horrified at a suggestion to "massage" the data to overstate the change that had occurred. Yet, the use of these techniques has the same effect on an audience's recollection of the data.

Example 4c



Both two and three dimensional examples also suffer from an additional failing relating to the spacing of the x axis. Because the spacing between each figure is constant and each figure is progressively larger, the distances between the years, although of equal size, are not equally represented further emphasizing the most recent time interval.

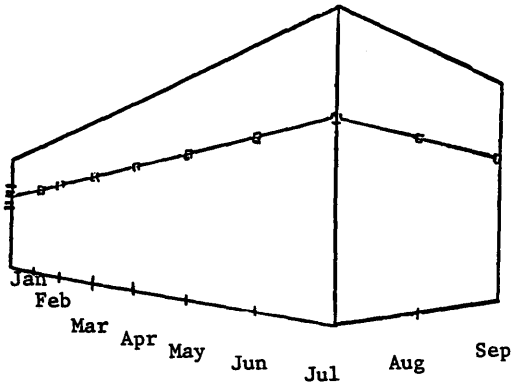
Example 5a shows another technique used to enliven graphics. In this situation, the analyst was attempting to illustrate the number of hardware failures that occurred prior to and immediately after switching to another hardware vendor's DASD. Note that the time the hardware change occurred is in the foreground. Through the use of perspective, the analyst was able

to convey the mistaken notion that the number of outages has fallen off since the hardware change, even though the number of outages did not change significantly.

Example 5a

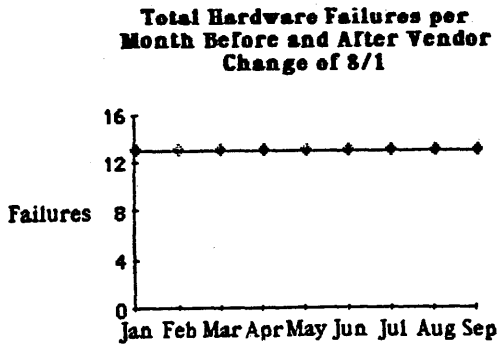
Total Hardware Failures per Month

Before and After Vendor Change of 8/1



Example 5b restates the data in a simpler fashion, and it is clear that there is no significant increase in number of hardware outages after the change of vendor. In conclusion, the use of perspective has little to offer the analyst interested in accurately portraying data.

Example 5b

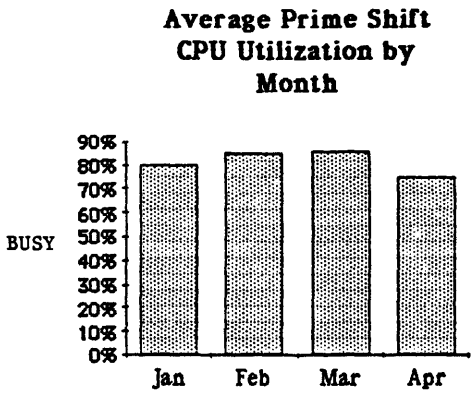


STANDARDIZATION

At times, an analyst will neglect the fact that value of a number can change over time. A common situation that plagues business in general relates to money and inflation. For example, if one were comparing the budgets for data processing from 1967 to the present, the increases in expenditures would be grossly overstated in excess of 300 percent because of inflationary effects. Thus, economists rely on using standardization to create constant dollars.

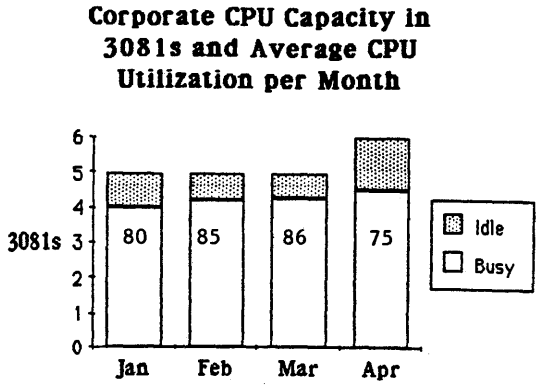
A more specific situation that can occur in data processing related to trying to represent available capacity when the configuration changes. In the following example, 6a, the data center is tracking average CPU utilization. At first glance, it appears that there was a drop in workload in April. The initial suggestion might be that due to vacations coinciding with Easter, the workload was down. However, what actually happened was that a field upgrade had increased the capacity of some of the processors, and, in fact, the number of service units processed actually increased. What is missing is some manner to indicate changes in the size of the base; in this situation, the number of standardized 3033 or 3081 processors. Perhaps the simplest technique would be to annotate the April entry to indicate the increase in capacity.

Example 6a



An alternate implementation, represented in example 6b, is to use a histogram, the height of which indicates the total CPU capacity of the center and employs stack charting to represent the proportions idle and busy. In this example, it is possible to jointly track the capacity of the data center, the extent to which the CPU is in use, and aggregate CPU use.

Example 6b



One final alternative is to present the rather simple table in example 6c.

Example 6c

Corporate CPU

Capacity in Equivalent 3081s

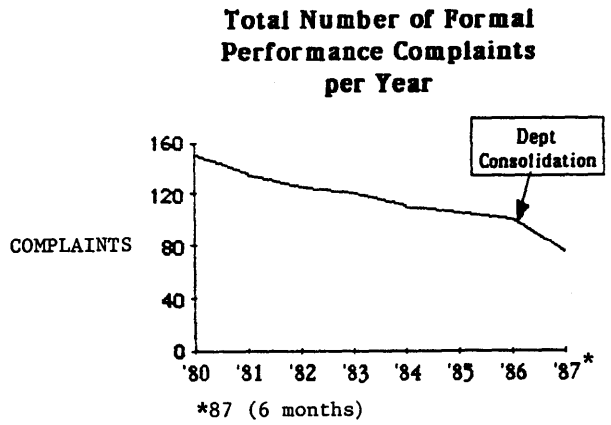
and Average Prime Shift

CPU Utilization by Month

	Jan	Feb	March	April
Total Capacity in Equivalent 3081s	5.0	5.0	5.0	6.0
% Idle	20	15	14	25
% Busy	80	85	86	75

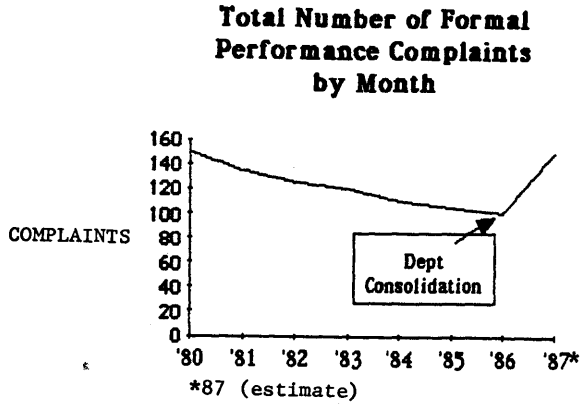
Another error can occur that mixes the effects of changing the scale of the horizontal axis with raw count data.

Example 7a



In example 7a, the analyst has plotted the number of customer complaints about performance, particularly emphasizing the number of complaints under the long regime of his predecessor as compared to those under his own relatively short reign. On first blush, the current analyst has had a noticeable effect on reducing complaints, much more so than had the earlier analyst. However, the situation is not as rosy as it appears, since the data for the prior analyst represents counts for 12 month periods while the last interval represents the past six months. Given no other additional information, it is likely that the number of complaints for the remaining six months will equal the prior six months, and, when this adjustment is made for the current year, the graph is as presented in 7b.

Example 7b

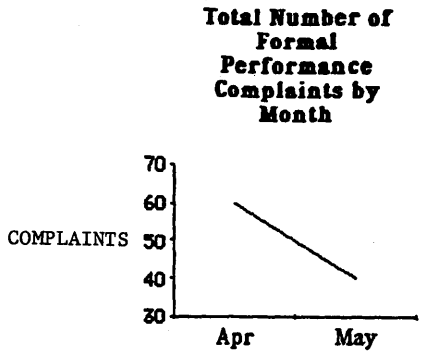


Obviously, the new analyst is not yet as effective as his predecessor.

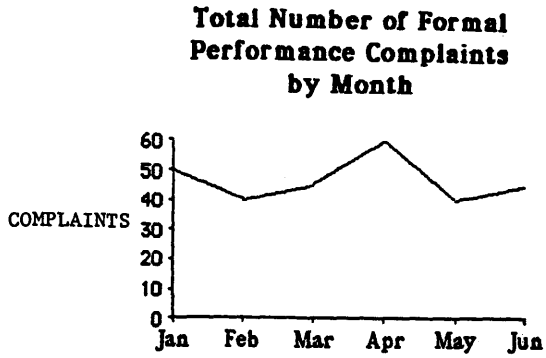
DATA INTEGRITY OR PRETEST/POST-TEST

Often in performance analysis, we must make evaluations of the effectiveness of some system or operational change on the net performance of the data center. At its simplest, there is a pretest, we make the change, we test again, and compare the results. The fallacy that can creep in is procedural in nature and can be reflected in simple graphics used to present the results. Example 8a attempts to summarize the effects of a 322 exit to prevent run-away applications by automatically canceling a session that uses excessive amounts of CPU. There appears to have been a rather significant reduction in the average CPU time per session, although to a large extent this reduction is over-represented by starting the Y axis at 30 rather than 0. However, the analyst failed to consider if the reduction in CPU time was greater than the normal variation in CPU time per session. By plotting several months prior, as in example 8b, it is clear that the current level of CPU consumption after implementing the 322 exit is not significantly different than the months immediately prior. This could also be validated with a traditional test of significance. In sum, whenever evaluating the effect of a system change on performance, it is necessary to evaluate more data than just the two time periods immediately adjacent to the point of change to assure that the change is significantly greater than the normal variability.

Example 8a



Example 8b

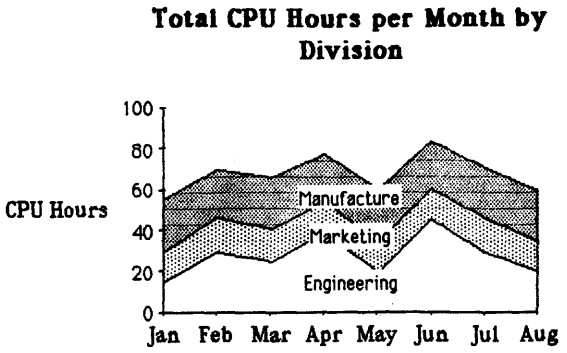


DATA ORDERING

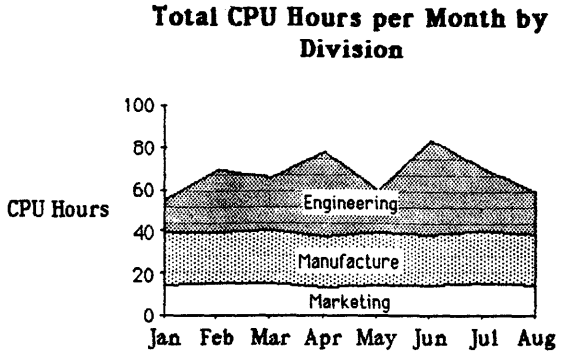
Care must be taken in the order in which data is presented in a chart. One type of chart that is particularly prone to this type of problem is the multiple strata surface chart. This form of rectilinear coordinate chart portrays the component parts of a total that can be expressed either in absolute numbers or in percentages.

Examples 9a and 9b show two very different looking charts. The first suggests very unstable data for all three divisions. In contrast, the second reveals that only the Engineering Division's use of the CPU has been erratic over time, while the other divisions have had rather stable CPU use. However, both of these examples are based on the same data with the only difference being that CPU use by the Engineering Division is presented at the bottom of the chart in the first example and at the top of the chart in the second example.

Example 9a



Example 9b



When using a multiple strata surface chart it is important to recognize that the basis for reading the values of the strata is the distance between the plotted lines and not the distance of the lines from the baseline. In this type of chart, only the total and the bottom strata can be read from the baseline. Therefore, the point being made by the analyst is the contrast in the fluctuations in one strata relative to the consistency in the others, the strata exhibiting considerable fluctuations should be placed at the top. On the other hand, if the major point being made is that the overall change is due to the monotonic change in one of the strata, that strata should be placed at the bottom most strata.

PRESENTING THE DATA CLEARLY

There are several practices that can make the point of a graphical presentation easier to determine: avoid pie charts; avoid the moire effect; use care if color is employed, minimize unnecessary clutter.

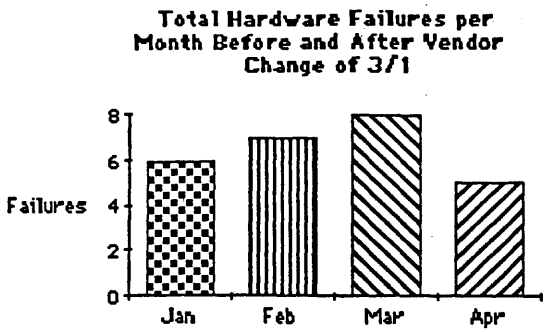
AVOID PIE CHARTS

Although commonly used and produced by virtually every graphics package, pie charts are generally frowned upon by statisticians. One of the general failings of pie charts is the lack of any explicit scale to evaluate the size of a particular slice. This is the reason that so many pie charts have categories explicitly labeled with their values. Unless the analyst is attempting to reinforce the notion that some component of the whole is equal to half or perhaps one quarter, there is little to recommend the use of a pie chart. Probably the only picture worse than a pie chart is a page full or pie charts with the expectation that the audience will compare one pie chart to another. Pie charts are a terrible choice where the intent of the analyst is to compare and contrast multiple percent distributions. The only singularly prevailing reason to use a pie chart in a presentation is that your boss insists upon it. A table or a 100 percent bar or column is almost always preferable. If you must use a pie chart, restrict its use to no more than four categories, and never use multiple pie charts to illustrate differences in distributions.

AVOID THE MOIRE EFFECT

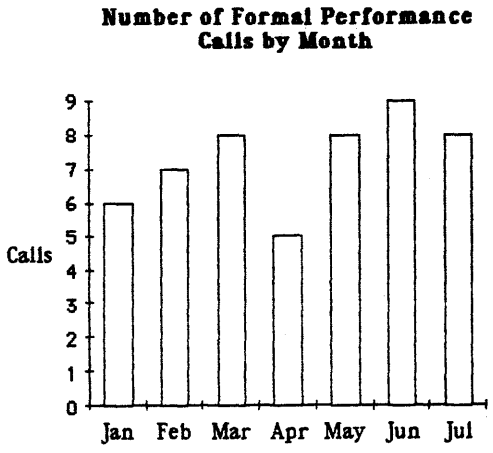
This effect describes the situation where a picture appears to pulse, vibrate or move. When such an effect is present in a graph it distracts the audience from the message of the graph. Essentially, the Moire effect occurs when patterns are presented with extreme contrast as depicted in example 10a. This effect is seen with the use of hatching to distinguish subgroups, and has become increasingly prevalent, particularly with non-color graphics. There are two remedies which can be used singularly or in tandem: preferably use words to label the subcategories; and avoid stark contrast by using shades of grey.

Example 10a



The second situation where the Moire effect can occur is depicted in example 10b. In this situation, the graph appears to move because of the close equal spacing of the boundaries of the histogram and their separating spaces. In this situation, the problem can be avoided by spacing the bars closer or farther apart than the width of the bars.

Example 10b



USE WITH COLOR

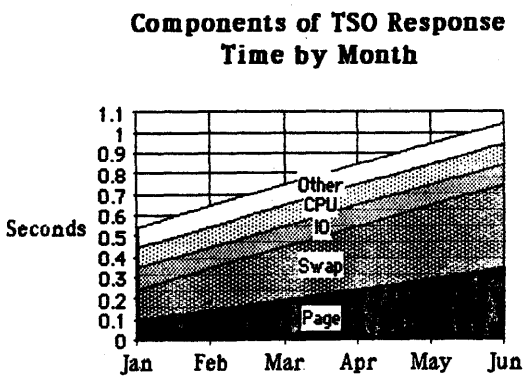
The use of color can be a two edged sword. If you have access to color graphics, color can be used to avoid the Moire effect caused by hatching. However, the inappropriate use of color can make an otherwise admirable graphic unreadable. Particular care needs to be made with slides using a dark background, such as when you are attempting to simulate the appearance of a

3179 color graphics terminal. Among the worse colors to use for text are the dark colors, like a deep red or blue, which do not project well at all. Instead use the light pastels to assure that the picture can be read. Since one male in eight experiences some degree of color blindness, the colors green and red should be avoided for important contrasts in the graphics themselves.

MINIMIZE THE CLUTTER

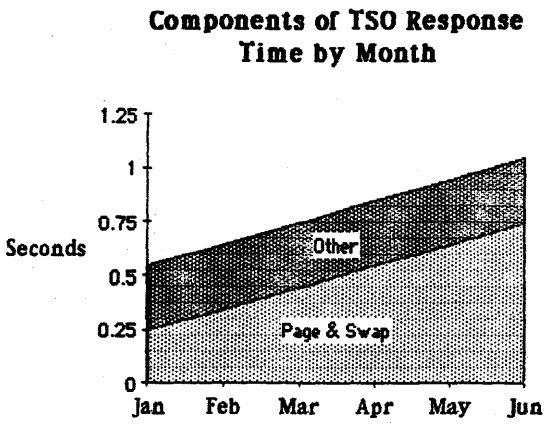
It is important to distinguish between graphics used for data analysis and those used for presentations. When graphics are used for analysis, the skill of the analyst can usually see through the excessive data and presentation clutter to the underlying information. In order to provide useful presentation graphics, the analyst must discern the message he intends to convey with the graph, eliminate or consolidate unimportant data, remove grid marks, and simplify the scaling so the information is not overwhelmed with the supporting structure of the graphic. In example 11a, we see an illustration of a rough contour chart which is reviewing trends in the wait reasons for TSO response time.

Example 11a



It suffers from too many insignificant wait reasons, too many scale annotations, and the clutter of grid marks, which are only useful for drawing a graph by hand. The surface chart in example 11b focuses on conveying that the growth in TSO response time is due primarily to a growth in the time spent paging and swapping. It accomplishes this task by first consolidating the less significant wait reasons, and presents the page and swapping delays adjacent to the horizontal axis to facilitate comparisons of the growth in page and swapping delays over time with the change in overall response time.

Example 11b



CONCLUSION

The use of pictures to represent data has become increasingly prevelant in management level presentations in data processing. This is due largely to the perception that the use of charts and graphs makes a presentation both easier to understand and more professional in appearance. Because of increasing use, it is more important that a graph accurately and clearly represent the data at hand. To this end, a variety of common graphical errors have been highlighted, and some suggestions have been made to clarify the use of pictures to represent data.

REFERENCES

[1] Dickson, Gary W., Geraldine DeSanctis, and D. J. McBride, "Understanding the Effectiveness of Computer Graphics for Decision Support," Communications of the ACM, vol. 29, 1, (1966) pp. 40-47.

[2] Schmid, Calvin F., and Stanton G. Schmid, HANDBOOK OF GRAPHIC PRESENTATION, New York: John Wiley and Sons (1979).

[3] Tufte, Edward R., THE QUANTITATIVE ANALYSIS OF SOCIAL PROBLEMS, Reading, Mass.: Addison-Wesley (1970).

[4] Tufte, Edward R., THE VISUAL DISPLAY OF QUANTITATIVE INFORMATION, Connecticut: Graphics Press (1983).

[5] Tukey, John W., EXPLORATORY DATA ANALYSIS, Reading, Mass.: Addison-Wesley (1977).

AN IMPLEMENTATION OF A GRAPHICAL MANAGEMENT INFORMATION SYSTEM USING SYMPHONY

DYNA WILSON
CoreStates Financial Corporation

ABSTRACT

Hard copy monthly status reports for Performance and Capacity Planning can never be timely given the nature of their construction. Time is required after month end for mainframe data extraction. Data reduction must be accomplished to make the data relevant. And then the data must be made understandable to non-technical users and prepared for presentation to management. Just when the report is being completed, we discover a missing data item. We wait. Finally, the report is in final form and our job is done. We send the completed report to central word processing for last minute corrections, duplication and distribution. They give it three day turnaround.

The monthly status report is out in three weeks. Not a bad month.

At CoreStates, we have implemented a PC based graphical application which enables key managers to view performance and availability data as it becomes available at month end, long before the monthly status report is completed and distributed.

An implementation of a graphical Management Information System for Capacity Planning and Performance using Symphony graphics is described.

1. INTRODUCTION

CoreStates Financial Corporation is the 36th largest bank holding company in the United States when measured in assets. However, we are in the top ten of the largest VISA issuers, international funds transfer banks, and check processors. We are the proprietors of MAC which is either the first, second or third largest shared ATM network in the country, depending on which measure you prefer. Each day, we process over a million IMS transactions, 100,000 CICS transactions and 250,000 TSO transactions. Nearly 200,000 batch jobs are required each month. This is accomplished with 5 IBM main-frames (2-3033, 3-3081) and 180 gigabytes of DASD. More than 8000 terminals, ATM's and Point-of-Sale (POS) devices are supported on our network via 37x5's, using VTAM and NCP, on some 180 leased lines. Additionally, 60 Tandem processors are supported as switches for MAC, POS, International and Funds Transfer operations.

The Performance and Capacity Planning Group at CoreStates is tasked with generating, accumulating and analyzing performance statistics across all business areas for all systems. We are tasked with enduring as opposed to reactionary performance management, bottleneck identification and prediction, and strategic capacity planning. As part of a mobile reserve, we are called upon to provide data for reactionary problem resolution and serve as translators for management.

To give us a firmer foundation for our translation duty, we began publishing a monthly management briefing book about three years ago. It contains performance and availability statistics presented in graphical and tabular form for the hardware, software and communication components which support key business elements within the corporation. Appropriate historical information is presented with each statistic to define perspective and demonstrate trends. The briefing book also provides a vehicle to transmit the number and type of installed equipment, financial performance of various groups or products, and staffing levels. In accordance with its intended purpose of presenting concisely and at a high level those issues of current interest to management, its contents are free to evolve in concert with corporate business directions.

This year, we have created an interactive Graphical Management Information System (GMIS) so that the performance and availability graphs found in the management briefing book are now available in the managers' offices within a few days of month end. Both the hard-copy management briefing book and GMIS are effected using Symphony software which is licensed by Lotus Development Corporation. Table 1 lists the graphical reports which are included in GMIS.

2. SYSTEM OVERVIEW

GMIS is an interactive macro driven graphical display system which uses the Symphony menu interface. The customized menus, appearing on the Symphony command line, are a two level hierarchy and can be traversed via single keystrokes. As each menu choice is highlighted by the cursor, a description of the action is displayed on the line above. The main menu is responsible for exiting GMIS, invoking a continuous slide show or selecting a submenu. The entry screen displaying the main menu in the command line is shown in Figure 1. Each of the second level menus gives choices of specific graphical displays as well as a return to the main menu. The menu hierarchy is depicted in Figure 2. To display any of the 28 graphs available, the user first displays the submenu that contains the graph title by entering the first letter of the submenu name while in the main menu. At the submenu display, the user enters the first letter of the graph title to display the graph. Alternatively, the user may move the cursor to highlight any entry and enter return. To terminate any graphical display, the user enters any single keystroke and is returned to the submenu from which the graph was invoked.

GMIS can be loaded and invoked by a single keystroke from the DOS prompt. However, those managers whose PC's support multiple sessions usually choose to leave GMIS loaded and active in one window. GMIS can also be exited and provide a return to DOS with a single keystroke from its main menu.

FIGURE 1
GMIS ENTRY SCREEN

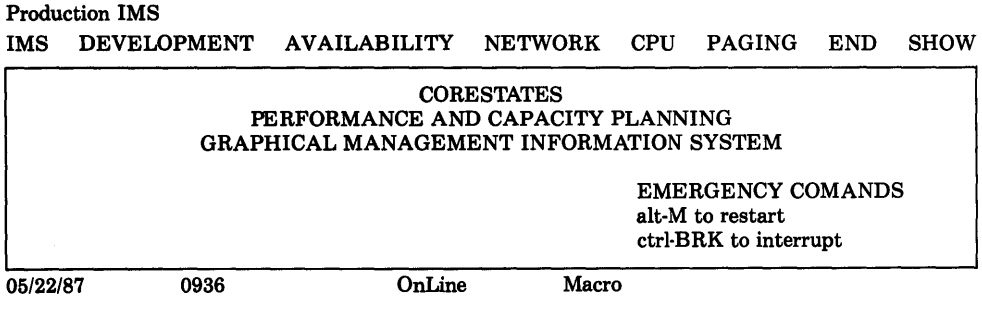


TABLE 1
GMIS GRAPHS BY TYPE

PERFORMANCE

IMS Message Volume of Top Applications for 24 Months

IMS Daily Average Message Volume by Month for 12 Months

IMS Average Response Time per Message by Month for 12 Months

Non-MAC Enter Key Daily Average Volume by Month for 12 Months

Non-MAC Enter Key Average Response Time per Enter Key by Month for 12 Months

TSO Daily Average Transaction Volume by Month for 12 Months

TSO Average Response Time per Transaction by Month for 12 Months Batch Workload by Type by Month for 12 Months

Percent CPU Utilization by Hour of the Day for 3 Months (for each processor)

Paging Rate by Hour of the Day for 3 Months (for each processor)

AVAILABILITY

IBM Mainframes Percent Availability by Month for 12 Months

Tandem Complexes Percent Availability by Month for 12 Months

Top Five Application Availabilities by Month for 12 Months

Database Management System Availabilities by Month for 12 Months

Voice Switch Availabilities by Month for 12 Months

Voice Tie Lines Availabilities by Month for 12 Months

Domestic Data Networks Availabilities by Month for 12 Months

International Data Network Availabilities by Month for 12 Months

Audio Applications Availabilities by Month for 12 Months

Audio Equipment Availabilities by Month for 12 Months

FIGURE 2
MENU HIERARCHY

MAIN MENU ENTRY	SUBMENU ENTRY
IMS	Month Daily Enterkey Transresponse Keyresponse Return
Development	Volume Time Batch Return
Availability	IBM Tandem Applications Databases Voiceapp Audioequip Return
Network	Domestic International Tielines Switches Return
CPU	A-3081 B-3081 C-3081 Transit (A-3033) MAC (B-3033) Return
Paging	A-3081 B-3081 C-3081 Transit A-3033) MAC (B-3033) Return
Show	
End	

3. SYSTEM ARCHITECTURE

GMIS is supported by Symphony v2.1 and is run on MS-DOS compatible personal computer with 256K internal memory and graphics board. It is also supported by Symphony v2.1 on the IBM 3270PC with 640K internal memory and graphics board running the 3270PC Control Program.

GMIS is a single self-contained Symphony worksheet file. The worksheet file is logically divided into four sections: Entry Screen; Main Menu Macros; Submenu Macros; and Data. The graphs are generated from Symphony graph configuration files and displayed via calls to predefined graph windows. The graph configuration files and window definitions are stored internal to the Symphony worksheet file. The Main Menu Macros and the Submenu Macros serve to display the command line menus and descriptions, automate calls to specific windows, and control program flow.

4. INITIALIZATION

GMIS is loaded from the DOS prompt by entering an "S". This instructs DOS to run a batch file named S.BAT which is stored in the root directory. The batch file as exhibited causes the correct color driver to be loaded, changes the current directory to SYM and invokes Symphony from that directory. When Symphony is exited, it changes the current directory to the root directory.

```
S.BAT
  mode co80
  d/sym
  symphony
  cd/
```

Two settings must be preset and saved in the Symphony configuration file to cause GMIS to self-load and self-invoke. First GMIS must be specified in the Symphony configuration file as the worksheet to be loaded automatically. This is accomplished by selecting CONFIGURATION from the SERVICES menu and entering GMIS at the AUTO prompt. Second, the macro which initiates GMIS must be specified as the macro to be executed when GMIS is loaded. This is done by selecting SETTINGS from the Symphony SERVICES menu and entering the initial macro name at the AUTO-EXECUTE prompt.

5. IMPLEMENTATION

Each menu appearing in the command line is precisely structured in a portion of the spreadsheet that is labelled with the menu name. In this structure, each menu choice is entered horizontally, one per cell, with no embedded blank cells. The first cell beneath each menu item contains the description which displays when that menu item is highlighted. Below this, Symphony macro commands are entered vertically with the first blank cell signalling the end of control. The macros which implement the following functions are described in detail: branch to submenu; display a specific plot; return to main menu; continuous slide show; ending GMIS; and error handling.

Branch to a Submenu:

```
NETWORK
Data, Audio, Voice
(menubranch menunet)
```

The menu choice appears in capitals, the description of the action which is displayed when this choice is highlighted appears in the second line and the program control is passed to an area labeled "menunet".

"Menunet" contains the submenu choices for the data, audio and voice network graphs.

Display a Specific Plot:

```
DOMESTIC
Domestic Network Availability
(paneloff)
(services)wunetdom
(?)
(services)usmain
(home)
(panelon)
(menubranch menunet)
```

Lines 1 and 2 display the submenu choice and description. Line 3 causes the command line display to be suppressed during macro execution. Line 4 calls the SERVICES functions of Symphony to display a previously defined window named netdom. In this case, netdom is a graph window which contains a 12 month domestic network availability graph. The (?) in line 5 allows the macro action to be suspended while the user is viewing the graph. When the user enters a keystroke signalling that the user is done, the GMIS introduction screen, located at the home position of the main window, is displayed. Command line display is enabled in line 8 and control is passed to the area labeled "menunet" in line 9. "Menunet" is the area containing submenu choices and macros for the network category. At this point, the command line will contain the customized menu for the network category and the user is free to make another choice.

Return to Main Menu:

```
RETURN
Return to Main Menu
(branch /m)
```

Returning to the main menu from any of the submenus is accomplished through a straight forward branch to the initial macro which displays the main menu.

Continuous Slide Show:

```
SHOW
Continuous Graphical Display
(window)
(wait @now+@time(0,0,10))
(branch show)
```

Lines 1 and 2 display the menu choice and description. Line 3 causes Symphony to display the first previously defined window it encounters. The wait command in line 4 causes execution of the macro to delay for 10 seconds. At the end of 10 seconds, a branch to the area labelled "show" is completed. "Show" is line 3 of this macro. At this execution of the window command, Symphony displays the next previously defined window and in the case of the last defined window, wraps around to display the first defined window. Thus, every 10 seconds, a new graph appears on the screen and continues until the user effects an interrupt via a cntrl-brk.

Exit from GMIS:

```
END
Exit to DOS
(services)ey
```

A call is made to the SERVICES command to exit Symphony. Control is returned to the S.BAT file which returns the user to the root directory of DOS.

Error Handling:

```
(esc)
(paneloff)
(services)wumain
(home)
(panelon)
(branch /m)
```

If an error is encountered during the execution of any macros, control is passed to the above macro. Line 1 causes the error to be cleared. Lines 2-5 suppress the command line display and return the window display to the introductory screen. Line 6 transfers control to the area labelled "/m" which reinitializes GMIS. To effect the branch to the error handling routine, it is necessary to label its location (eg., /e) and include the following command in the initialization macro: (onerror /e). Note that when the user ends the slide show via cntrl-brk, GMIS is reinitialized through this error handling routine.

6. SYSTEM SECURITY

Protection of the system against inadvertent or intentional change is afforded with a two step process. First, GLOBAL PROTECTION is set to yes from the SETTINGS selection appearing in the SERVICES menu for the entire worksheet. This prevents updates to any cells. To ensure that no one changes this setting, the worksheet must be password protected. This Lock is entered from the SECURITY option of the SERVICES SETTINGS menu.

Another command sequence in Symphony allows all the macro and/or data areas to be hidden from the user's view until the password protection is unlocked and the setting manually changed. To make the macro and/or data areas non-viewable, set the OTHER option in the MENU FORMAT selection to HIDDEN and select the entire macro and data range. This must be done while the worksheet is free from password protection and the Global Protection has been released. Once accomplished, the password is activated as described previously. The password is then required to allow the users to view the protected macro and data areas.

7. MONTHLY MAINTENANCE

At month end, data is entered into some half dozen Symphony worksheets for analysis and preparation of the hard copy Management briefing book. As each of these worksheets is updated, the information pertinent to GMIS can be conveniently copied to the single GMIS worksheet, overlaying only the portion of the GMIS worksheet which is being updated. Thus, for example, if for some reason batch performance reporting is delayed, GMIS could still be distributed in a timely manner with updates to all the graphs except the batch performance. The batch performance would remain at the prior month level.

8. CONCLUSION

GMIS has been embraced by CoreStates Managers as a vehicle for timely monthly reporting. They enjoy its speed and user friendly menu driven format. As such, GMIS has proved to be a low-cost, low-maintenance solution to performance standard reporting.

THE NEED FOR A NETWORK MONITOR

Phil Bodley
TSB (England & Wales) Bank plc

ABSTRACT

This paper describes why a non-IBM user with large mainframe installations and an extensive network decided to buy an independent network monitor.

It starts with a horror story for capacity planners, but promises a happy ending.

Some network managers and capacity planners may draw comfort by realising how easy their lives are by comparison. Others may find some disturbing similarities with their own installations, which might prompt them to consider a similar solution.

1. PLENTY OF NETWORK CAPACITY . . .

A new central complex, new programs, and a new network were installed in 1980 to replace an earlier system. The replacement network served to connect the 950 high street branches of the bank with a new computer centre at Wythenshawe, a suburb of the northern city of Manchester. The capacities of the replacement mainframe and network were more than adequate for any peaks of traffic in the forseeable future; this was essential because all transactions in TSB branches are processed online and in real time i.e. whilst the customer is present (as they have been for many years).

The new central complex was powered by a Sperry 1100 mainframe, theoretically capable of processing 70 transactions per second.

See Figure 1 for a simple diagram of the network.

Each teller position on the counter of each branch was equipped with an intelligent Burroughs Workstation comprising processor, screen, keyboard, security card reader, passbook printer, and data cassette for program loading. Additional workstations were installed for "back-office" use, making some 4500 workstations in total.

Three or four neighbouring branches were multidropped onto a 2400bps line. Several such lines converged on a regional concentrator site containing between three and eight Burroughs CP9558 concentrator computers. There were 38 such concentrator sites housing 150 concentrators. Each concentrator handled the traffic from two of the incoming 2400bps multidrop "downstream" lines. If a concentrator failed then its downstream lines could be switched to adjacent concentrators by an intelligent matrix switch (IMS) operated remotely from Comms Control at Wythenshawe. Workstation transactions were logged by the concentrators before being passed up to the computer centre on 4800bps "upstream" lines of which there was at least one for every three concentrators.

All well and good.

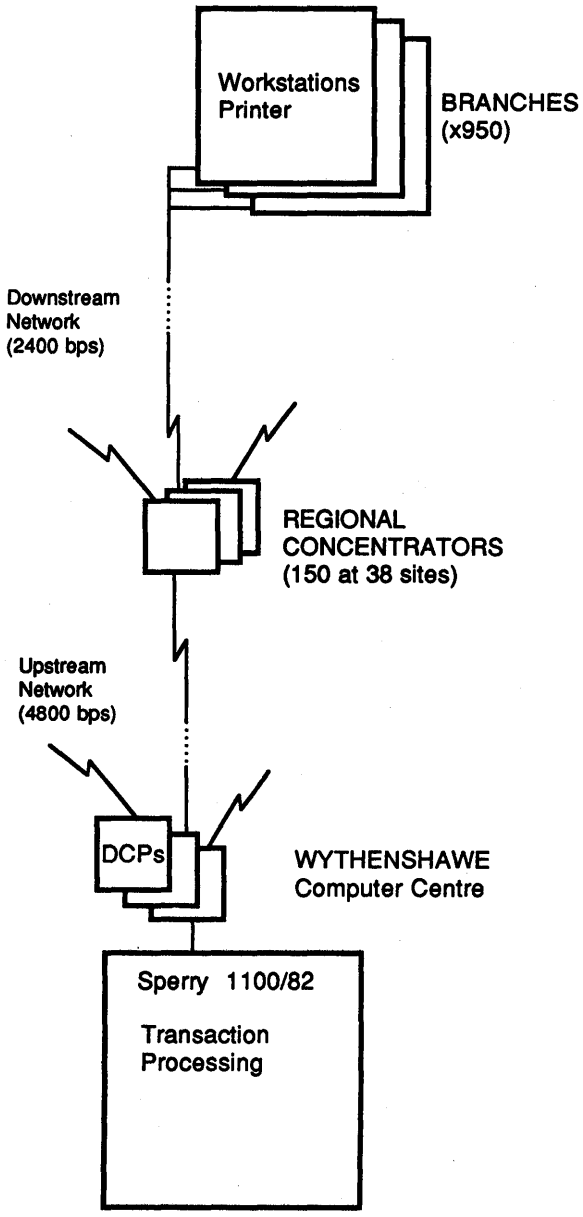


Figure 1. 1980 Network

2. . . . AND HOW TO USE IT UP . . .

Initially the traffic on the network consisted of the sort of basic banking transactions that one might imagine, covering, for example,

- deposits
- withdrawals
- enquiries for account balance and other details
- opening or closing an account
- amending a standing order

In addition, the concentrator (not the mainframe) provided some transactions to assist the teller:

- a list of transactions for the day (to help resolve any balancing problems)
- a menu of all possible transactions

The transaction traffic was confined to weekday prime shifts, peaking at around 10 a.m. and again mid afternoon, with Friday mornings generating the heaviest load. The replacement network had more than adequate capacity for these peaks.

Overnight the computer centre processed the transactions associated with clearing cheques, paying standing orders, and paying loan instalments. The branches needed daily reports of any exception conditions that arose overnight, such as cheques 'bouncing'. The network was soon being used to distribute those reports. The concentrators played the major part in this, by firing special enquiry transactions to the mainframe. The mainframe would respond to each of these transactions by returning a multiple-block response containing the report lines. On receipt of a particular report for one of its branches the concentrator would format the report and transmit it downstream to the printer in the branch. Quite a load on the network. Fortunately, report distribution for the day was designed to be completed between 0600 and 0800 whilst the network was otherwise idle.

By and by more branch reports were conceived and implemented. Good job the network and mainframe had the capacity before 0800, and capacity during prime shift to cover reports that had earlier fallen victim to paper wrecks, printer failure, or some other fate.

All the time the online system serving the branches was being enhanced, to provide such facilities as mortgage processing, foreign exchange, online customer details, credit scoring for loan applications, and many more. Some new facilities involved additional transactions, others increased the message lengths of existing transactions. Hence more prime shift workload for the network.

Then cash dispensers arrived, online real-time of course. So now the network had to support the additional workload of customers enquiring for their up-to-the-minute balance before withdrawing their cash!

The cash dispensers fronting onto the street were then supplemented by nine hundred customer-operated autotellers inside many of the branches. These machines not only allowed cash withdrawal, but also deposits, enquiries about standing order mandates and payments, printing of mini-statements and, most significantly, printing of complete up-to-the-minute statements. The additional prime shift traffic on the network caused by customers browsing through their standing orders or perhaps getting a statement printed 'nice and conveniently' can be imagined.

It is also not hard to imagine how the polling overhead and poll delays might have been increasing as all these extra devices were configured on the network. These weren't the only extra devices . . .

All this time the Bank's volume of business had been increasing. Some of this growth was steady, such as the upward drift in the average number of transactions per account per year, whilst other growth was explosive, such as taking on 0.75 million new cheque accounts in a single year. Naturally this led to much more network traffic, and also placed a strain on the existing population of workstations, so more were installed, further increasing the polling overheads.

At this point the mainframe was virtually collapsing under the weight, and had to be rapidly upgraded. Even before this could be done its overnight processing capacity became exhausted, so the creation of the all important daily reports ran late. Hence the transmission of the reports via the network to the branches could not always be completed, or on some black days not even started, before prime shift. Saturation. So when the branch staff found no reports, what was their natural reaction? To flood the system with enquiry transactions, firstly to request transmission of the missing reports, and when that was unsuccessful, to enquire for the very details from the database which the reports would have given them. Total saturation, relieved only by upgrading the mainframes and reverting report transmission to its normal early slot. Serious capacity planning got the boost it needed.

The extra workstations and the autotellers mentioned earlier were not the only aspects of change in the network configuration. It seemed forever volatile. Concentrator sites were closed or moved because the lease had run out, or because of redevelopment, or because of the threats of local civil unrest, or whatever. New branches were opened, some branches were closed. Branches connected to one concentrator site had to be switched to another to fit in with company reorganisations and regional boundary changes. Thus traffic rates would be waxing through one node of the network, and theoretically (!) waning in another.

One of the more significant changes to the configuration stemmed from adding to the network the 230 branches in London and the South East corner of England, and in the Channel Islands. For historical reasons they had previously been served by a separate online system which was incompatible with the Sperry-based system used elsewhere. A fully integrated system was a high priority goal. A concurrent requirement to provide complete disaster standby for the branch banking system was satisfied by the construction of a functionally identical computer centre at Milton Keynes (a new city 70 miles northwest of London), linked to Wythenshawe by very high speed lines. It was to this second centre that the 230 branches were connected, via thirty new concentrators. The branches retained their existing terminals until they could be replaced with the standard Burroughs workstations. Good job this was done quickly. The old terminals were dumb, so the concentrators had to emulate all the functionality of the Burroughs workstation, and the downstream lines were saturated with screen images. Hardly suprisingly the response times were poor, until the branches received their new kit.

Once the new computer centre had settled down, some of the Wythenshawe workload was transferred to Milton Keynes. Thus eleven hundred branches became connected through 180 concentrators to both computer centres, but with each centre normally handling the traffic from its half of the country.

Still the configuration changes continued. Now, since part of the configuration data is held in concentrator software there was seen to be an advantage in using the network to download that software. Hence a File Transfer facility was developed, adding a workload to

the upstream network and concentrators (usually at offpeak times, fortunately).

Jointly implemented with File Transfer was the Burroughs Datascope facility. This enabled technical staff at the computer centres to remotely inspect the content of any file, including the system incident log, on any concentrator to help diagnose hardware or software problems. Another workload on the upstream network and concentrators, not necessarily confined to offpeak times.

Talking of diagnostics, the reliability of the cash dispensers is not only high profile, but also jeopardised by their electro-mechanical nature, and by fish-and-chips, curry sauce, and other street hazards. To enable failed machines to be restored to service as quickly as possible it was considered essential to know the state of all autotellers all the time. This was achieved by implementing bi-directional status monitoring; the autotellers tell the mainframe if they are having a problem, and the mainframe can periodically probe those that seem OK to check that they are still there and feeling well. With 1,100 machines to mind this became yet another workload on both the upstream and downstream networks and the concentrators.

All the while the range of facilities available to the branch staff continued to expand. What a lot of transactions to choose from! Unfortunately screen formats and vetting procedures for each type of transaction consume memory in the workstations. Back in 1979, when the workstations were chosen, 96K seemed very generous. You've guessed: insufficient memory. The solution adopted was for the concentrators to hold the screen formats and vetting code for the less frequently used transactions, and then, when required, to download them to an overlay area in the workstation memory. Thus a potentially substantial additional workload was imposed on the concentrators and downstream network during prime shift.

And still the developments rolled on.

Soon came CHAPS (Clearing House Automated Payments System) enabling large value (£10,000+) money transmissions between TSB and the other banks. Not high volume, but still an additional network workload.

Burroughs B25 micros were introduced to administrative centres and certain branches. These posed a threat to the system because they are capable of firing off a whole volley of online enquiry transactions at the touch of a button, impacting both upstream, concentrator, and downstream.

Software and communications links were put in place to allow TSB Scotland and TSB Northern Ireland cash card holders to use the TSB England & Wales cash dispensers, and similarly to allow Trustcard (VISA) holders to be advanced cash. More traffic.

A recent development under trial permits the use of Sperry's Mapper (4GL) from designated terminals. Another potentially heavy workload.

There is much else afoot, including reciprocal autoteller arrangements with other banks. The good news: our customers will use their autotellers, so less network traffic for us. The bad news: their customers will use ours . . . . and there are more of them!

Figure 2 shows the latter-day shape of this heavily loaded network.

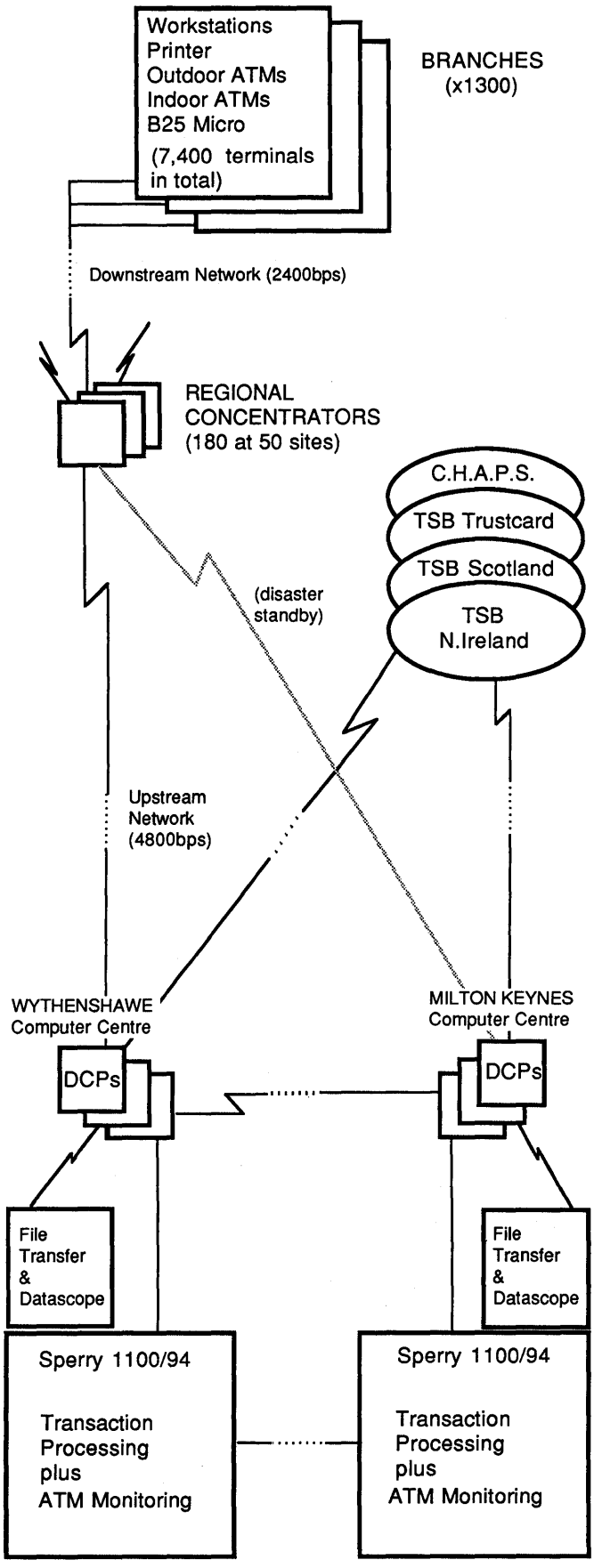


Figure 2. 1987 Network

3. . . . WITHOUT KNOWING IT!

Of course we knew about the escalating workload on the network, we could hardly fail to notice it!

The main measurement tool at our disposal was the mainframe. All transactions arriving at the mainframe are logged, and several programs have been developed to analyse the transaction traffic. For example, it is possible to plot the number of transactions originating from each downstream line of each concentrator during any chosen period.

Question: What other measurement tools were available? Answer: Not a lot.

The front end processors (Sperry DCP/40s) collected statistics about each upstream line, such as number of messages received/transmitted and number of characters received/transmitted. No tools were provided for retrieving, managing and analysing the statistics (no SAS on Sperry) so it was not until recently that they became of any use whatsoever.

We occasionally resorted to datascope line monitors to give us glimpses of the traffic. We even used stopwatches to time samples of transactions from a workstation or two (or relied on branch complaints). Such scraps of knowledge gained by very labour intensive means are so localised, geographically and in time, as to be quite unsatisfactory.

So we were left with significant unknowns about the activity in our network:

a) Little knowledge of the line loadings of each of the sixty upstream lines until the recent analysis of the DCP Statistics enabled estimates to be made.

b) Little knowledge of the contribution of the upstream to overall response time.

c) Little knowledge of the impact of initiating a file transfer from the computer centre to a concentrator, or of retrieving diagnostics from a concentrator.

d) Less knowledge of the behaviour of the 360 downstream lines, apart from number of transactions reaching the mainframe.

e) Very little knowlege of the impact of tellers requesting information, such as transaction logs or transaction menus, from the concentrators, or of how often they use these facilities.

f) Very little idea of the scale and impact of workstation program overlays being called up from the concentrator.

g) Very little idea of downstream line loadings, poll overheads and delays, retransmission overheads, contribution of downstream to overall response time, or contribution of concentrator to overall response time.

h) Very little idea of the impact on the network of the different transaction types.

i) Very little idea of how the network behaves during reports transmission.

j) Very little idea of the impact of autoteller health monitoring, or of CHAPS transactions, or of Mapper traffic.

k) Very little idea of the likely impact of adding additional devices to the poll trains, or of lengthening messages, or whatever, due to lack sufficient data to support modelling.

The knowledge that one doesn't have the knowledge tends to breed caution! Hence it has been policy to keep the workload fairly evenly balanced across the network, and to encourage sensible and economical use of the system.

Fortunately, the level of service provided to the users seems to have weathered most of the storms of change and growth described earlier.

"But for how much longer!" cry the capacity planners.

"But for how much longer!" cry the network administrators and controllers.

"But for how much longer!" would the users cry, if they knew what we know we don't know . . .

. . . and they probably will, because this year we are implementing service agreements between the user divisions and ourselves in Management Services.

4. FOCUSSING IN

You can't capacity plan the future if you don't know where you are now.

Similarly, you sign service agreements with your sponsors at your peril if your measurement and forecasting capability is suspect.

Clearly much, much more data on the network performance and workload was essential.

Perhaps the most obvious place for network measurement to be developed was in the concentrators, since they alone had a view of the otherwise quite separate upstream and downstream networks. If only they could have carried out round the clock recording of all measurements, for later transmission to, and analysis on, the mainframe! No chance. No suitable software was available from the supplier. Neither could any reasonably be developed in house given the limited capacity of the concentrators, the complexity of the work they are already carrying out, and the fact that all the development personnel skilled in Burroughs CMS/MPLII were fully stretched keeping up with the torrent of change.

The idea of an independent network monitoring system became increasingly attractive. Something that could be bought off the shelf, wheeled in, probed up to the lines, and left to collect data which it would then analyse and display in a wonderfully friendly format. Even better if it could then model proposed changes.

Just one or two problems.

Fine, if you've got a single level star network radiating from your centre, so that the monitor can be centrally sited. Unfortunately most of our lines (the downstream ones) are anywhere but near the computer centres.

Fine, if your protocol is 2780, 3270 or SNA. Un(?)fortunately not everywhere is an IBM site. Our protocols are Sperry HDLC (four-wire full-duplex but in Normal Resonse Mode) on the upstream, and Burroughs Poll/Select (half duplex) with Group Polling on the downstream; and they are, er-humm, slightly modified to give an extra high level control of the link, non standard multiblocking, . . .

Fine, if your transactions arrive in the same shape as when they left the terminal. Some of our transactions never get past the concentrators, and those that do are reformatted and addresses are stripped off so that only the concentrator software knows which device they came from.

Fine, if your traffic isn't cluttered up with unsolicited messages checking the wellbeing of autotellers, or carrying broadcasts to branches, or triggering special concentrator activity.

All-in-all, fine if your network is not TSB's!

With our shopping list of requirements and our purse we optimistically set off to the shops, only to find the shelves surprisingly bare. The supermarkets had no such departments, so we had to turn to the corner shops. Few of them had much to offer, even in plain, let alone fancy wrappers. Well there's nothing like having a short list to help you draw up a short-list.

It was clear that special protocol analysis would have to be developed by any potential supplier. It was also clear that to continuously monitor the entire network would require masses of such kit out near the downstream lines.

The short-list shortened itself further when one supplier decided it would only be worth doing the former (special development) if we committed to the latter (masses of kit).

Our purse was small; we were budgetted for only a modest system to start with. It would therefore have to be portable so that we could monitor the upstream and downstream lines for perhaps a week or two at one regional concentrator site before moving on to the next.

Eventually an order was placed with the only remaining supplier willing and able to meet our requirements at a price we could afford.

The system is a Dynatech PRISM capable of simultaneously monitoring twenty lines (upstream, downstream, or a mixture). It can measure at line, controller, or device level:

a) Response Time

- Average, Maximum, Std Deviation, No. over threshold

b) Host delay (ie upstream of monitoring point)

- Average, Maximum, Std Deviation, No. over threshold

c) Average Poll Delay

d) Number of Transactions

e) Percentage Line Utilisation

- Overall, Data, Protocol overhead, Inactive

f) Line Down Time

g) Line Idle Time

h) Retransmissions - Transmitted, Received

i) No. of Messages - Transmitted, Received

j) No. of Characters - Transmitted, Received

k) Total Number of Data Characters

l) Number of Polls

m) Number of No-Response-to-Polls

n) Number of Devices Active

Furthermore it claimed to be able to analyse the traffic down to individual transaction type. This was seen as one of the most important features of the system because of the wide variety of traffic, particularly on the downstream lines, and our lack of analysis of that traffic.

Once the PRISM system is fully operational it is hoped that

(i) the most threatened areas of the network can be analysed first, and a prognosis delivered before the service levels suffer.

(ii) it will serve as a "toe in the water" on network monitoring.

Thus we will start to form a clearer picture of the 1987 traffic on the 1980 network . . .

. . . <u>just as we are about to start replacing it with a 1988 network</u>!

5. <u>SUMMARY</u>

Piling the workload onto a finite capacity system of which one has only impaired vision is short sighted! We now see that . . . and hope the bifocals (ie upstream + downstream) we've bought will prevent us walking into anything unpleasant.

Ask yourselves: do YOU have an adequate picture of what is happening on YOUR network?

If not, how many years will pass before YOU identify in YOUR organisation THE NEED FOR A NETWORK MONITOR?

A Systematic Approach for Identifying Peak Periods

H. Pat Artis
Performance Associates, Inc.
Vienna, VA 22180

ABSTRACT: Basing capacity planning forecasts on peak periods is one of the most common approaches discussed by capacity planners. However, the identification of peak periods is not a trivial problem and is more often based on ***looking at the data*** rather than a defined procedure that can be automatically applied to large volumes of data. In this paper, we will examine statistical methods for identifying peak workload intervals based on a variety of measurements. The potential types of metrics that can be used for identifying peak periods include measurements of load, utilization, and service level.

1.0 Introduction

One of the most common problems presented to the performance analyst is the selection of peak intervals for modeling and/or workload forecasting.[1] However, the identification of these peak periods, if they do not occur at a regular time each day, is more of an art rather than a procedural approach that can easily be applied to large volumes of data and/or tuned to achieve more desirable results. As such, peak periods seem to be more illusionary rather than measurable and definable periods that do not require significant insight into the system's workload on behalf of the analyst to identify.

Perhaps the most common approach to the identification of peak intervals is the selection of time periods during which CPU utilization is very high. Historically, there have been two primary motivations for the use of CPU utilization to identify potential peak periods. They are:

- CPU utilization can be easily measured in a reliable manner,
- since the CPU is the most expensive single resource, management has historically viewed CPU utilization as a primary indicator of system capacity.

Although reasonable, these motivations are subject to question in today's environment. First, MVS/XA attempts to recruit (implemented by target MPL modifications made by the SRM) low priority (i.e., non-critical) work during idle periods in an attempt to maximize CPU utilization. However, the recruited workloads would be rapidly shed by the SRM in the event that some critical workload were to demand resources. Moreover, as the fraction of our workloads represented by on-line systems continues to increase, service levels become a dominate concern over CPU utilization.

Although one might argue that using service level values for the identification of peak periods offers a solution to the problem, arguments similar to those that were used to question focusing on CPU utilization can easily be hypothesized. Thus, any approach to the identification of peak periods must be multi-dimensional in nature, where weights can be assigned to various metrics in relation to the importance that they play in allowing the system to provide essential services and meet stated service objectives. Such multi-dimensional metrics used are called ***objective functions.*** Objective functions are most commonly used by optimization procedures to allow their searching algorithms to differentiate between alternative cases.

Defining an objective function for the identification of peak periods presents a number of problems. They include:

- the varying range of values over which each of the metrics is defined,
- the selection of the metrics used to define the objective function.

Each of these problems will be discussed in the following section.

2.0 Defining Objective Functions

The first problem presented to the analyst attempting to define a multi-dimensional objective function are the broad, and differing, ranges of values over which each of the potential metrics are defined. Perhaps the simplest way to address the problem presented by these varying ranges of values is scaling. For the purposes of this study, the author suggests that Z-statistic scaling be used to normalize each of the metrics. To Z-scale an observation, one simply subtracts the mean and divides by the standard deviation calculated for the metric. Z-scaling maps the vast majority of all observations over the range of -3.5 to +3.5, where a scaled value of zero implies that the observation was identical to the mean. Positive values are indicative of observations larger than the mean and negative values are indicative of observations smaller than the mean.

Once the potential metrics have been scaled, the analyst is presented with the question of which metrics should be included in objective function and what weights, if any, should be applied to the individual metrics. While the selection of the metrics is dependent on both the analyst's judgment and management's objectives for a specific system, potential metrics fall into three broad categories. They are:

- utilizations and other measures that are indicative of the hardware or the operating system's activities,
- indicators of load such as job and/or transaction counts,
- service level indicators such as response or turn around times.

In general, the scaled metrics describing any of these types of measurements can simply be added or averaged to yield an objective function value. However, in some cases, weights can be applied to the specific metrics to emphasize or de-emphasize their importance in the calculation of the objective function.

Although the use of weights is usually unattractive, there are specific cases where weights can be very useful. For example, consider the case of two TSO systems, a system for data center personnel (i.e., internal) and a second system for paying customers (i.e., external), sharing the same CPU. Although the service levels for the internal TSO system are of some concern, the revenue producing external system is management's primary concern. As such, applying a weight of 1/2 or 1/3 to the internal service metric would express its relative importance to the overall performance of the data center.

3.0 Case Study

To illustrate the approach to identification of peak periods discussed above, the technique was used to analyze hourly data collected for a CICS workload operating in an MVS/XA environment. Based on Detich's and Simpson's papers [2,3] that discussed the collection of key measurements to model CICS transaction response time, the following metrics were selected to characterize the hourly observations:

CPU Utilization: the percent of the CPU resource being consumed by all of the system's workloads. As the CPU utilization increases, it is more likely that a CICS transaction will encounter queueing delays as it competes for CPU resource,

Pages per Second: the load being carried by the auxiliary storage management system. Because of CICS's well known sensitivity to paging, this metric is considered to be a leading indicator of overall CICS performance,

Average Page Delay Time: the delay, in milli-seconds, experienced by a transaction when a page fault occurs,

Transactions: the number of CICS transactions processed during the hour,

Trivial Response: the average internal response time for trivial (i.e., transactions with low CPU and I/O requirements) CICS transactions.

The data collected for the CICS system during one prime shift interval is shown in **Figure 1.**

Hour	% CPU	Page/Sec	Avg PDT	Tran(K)	Trv Resp
8	68	113	4.5	53	0.41
9	77	140	4.9	137	0.65
10	95	241	5.7	187	0.87
11	94	237	6.3	181	0.92
12	72	155	4.6	71	0.49
13	86	201	5.1	172	0.80
14	97	229	5.2	184	0.85
15	96	233	6.1	175	0.91
16	87	198	5.1	162	0.75

Figure 1. Hourly System Measurements

A cursory examination of the data based on CPU utilization reveals that hour 14 might be the peak interval based on CPU utilization. However, other key indicators like average page delay time, CPU utilization, and response time do not indicate that the full resources of the system were stressed during the period. Using the approach discussed in the prior section, the means and standard deviation values were calculated for each of the metrics. These results are shown in **Figure 2.**

Hour	% CPU	Page/Sec	Avg PDT	Tran(K)	Trv Resp
8	68	113	4.5	53	0.41
9	77	140	4.9	137	0.65
10	95	241	5.7	187	0.87
11	94	237	6.3	181	0.92
12	72	155	4.6	71	0.49
13	86	201	5.1	172	0.80
14	97	229	5.2	184	0.85
15	96	233	6.1	175	0.91
16	87	198	5.1	162	0.75
STD	10.37	44.56	0.59	47.69	0.17
MEAN	85.78	194.11	5.28	146.89	0.74

Figure 2. Mean and Standard Deviation Calculations

By using these values to Z-scale the individual hourly observations, each of the potential contributing metrics can be scaled into a common range.

Using the scaled values, three different objective functions were evaluated. They were:

- an objective function based on CPU utilization, pages per second, average page delay, total transactions, and trivial response time,
- a four value objective function based on CPU utilization, average page delay, total transactions, and trivial response time,
- a three value objective function based on CPU utilization, total transactions, and trivial response time,

Each of these objective functions are discussed in the following sections.

3.1 Five Value Objective Function

The first objective function was defined as the un-weighted average of the Z-statistics of the five metrics discussed in the prior section. The objective function was calculated:

$$\text{Obj Fun} = \frac{\text{Z-CPU} + \text{Z-Page} + \text{Z-PDT} + \text{Z-Tran} + \text{Z-Resp}}{5}$$

The result of this objective function calculation is shown in **Figure 3.** The peak hour is denoted by the > and < symbols.

Hour	% CPU	Page/Sec	Avg PDT	Tran(K)	Trv Resp	
8	68	113	4.5	53	0.41	
9	77	140	4.9	137	0.65	
10	95	241	5.7	187	0.87	
>11	94	237	6.3	181	0.92<	
12	72	155	4.6	71	0.49	
13	86	201	5.1	172	0.80	
14	97	229	5.2	184	0.85	
15	96	233	6.1	175	0.91	
16	87	198	5.1	162	0.75	
STD	10.37	44.56	0.59	47.69	0.17	
MEAN	85.78	194.11	5.28	146.89	0.74	
						OBJ FUN
8	-1.71	-1.82	-1.31	-1.97	-1.89	-1.74
9	-0.85	-1.21	-0.64	-0.21	-0.51	-0.68
10	0.89	1.05	0.71	0.84	0.75	0.85
11	0.79	0.96	1.72	0.72	1.04	1.05
12	-1.33	-0.88	-1.14	-1.59	-1.43	-1.27
13	0.02	0.15	-0.30	0.53	0.35	0.15
14	1.08	0.78	-0.13	0.78	0.64	0.63
15	0.99	0.87	1.38	0.59	0.98	0.96
16	0.12	0.09	-0.30	0.32	0.06	0.06

Figure 3. Five Metric Objective Function

Based on this objective function, the peak interval occurs during hour 11, when CPU was only moderately loaded. Simply stated, the contribution of CPU utilization to the objective function was not an over riding concern when service related factors (i.e., page rate, average PDT, and response time) were factored into the calculation.

This objective function illustrates one potential concern of which the analyst must be aware. The use of two or more related metrics may place an undesired (and perhaps unanticipated) weight on one of the systems resources. In the five metric objective function introduced above, paging rate and average page delay are likely to illustrate a high correlation. To investigate this potential effect, a second objective function was evaluated using only one of the paging metrics.

3.2 Four Value Objective Function

The second objective function was defined as the un-weighted average of the Z-statistics of the four metrics. For this case, paging rate was removed from the objective function discussed in the prior section. The objective function was calculated:

$$\text{Obj Fun} = \frac{\text{Z-CPU} + \text{Z-PDT} + \text{Z-Tran} + \text{Z-Resp}}{4}$$

The result of this objective function calculation is shown in **Figure 4.** The peak hour is denoted by the > and < symbols.

Hour	% CPU	Page/Sec	Avg PDT	Tran(K)	Trv Resp	
8	68	113	4.5	53	0.41	
9	77	140	4.9	137	0.65	
10	95	241	5.7	187	0.87	
>11	94	237	6.3	181	0.92<	
12	72	155	4.6	71	0.49	
13	86	201	5.1	172	0.80	
14	97	229	5.2	184	0.85	
15	96	233	6.1	175	0.91	
16	87	198	5.1	162	0.75	
STD	10.37	44.56	0.59	47.69	0.17	
MEAN	85.78	194.11	5.28	146.89	0.74	
						OBJ FUN
8	-1.71	-1.82	-1.31	-1.97	-1.89	-1.72
9	-0.85	-1.21	-0.64	-0.21	-0.51	-0.55
10	0.89	1.05	0.71	0.84	0.75	0.80
11	0.79	0.96	1.72	0.72	1.04	1.07
12	-1.33	-0.88	-1.14	-1.59	-1.43	-1.37
13	0.02	0.15	-0.30	0.53	0.35	0.15
14	1.08	0.78	-0.13	0.78	0.64	0.59
15	0.99	0.87	1.38	0.59	0.98	0.99
16	0.12	0.09	-0.30	0.32	0.06	0.05

Figure 4. Four Metric Objective Function

As can be seen in the figure, the four metric objective function yields results similar to those developed in the prior section. That is, the peak interval still is identified as hour 11 in the study.

As a final step, an even simpler objective function was evaluated that excluded all virtual memory metrics. As a broad guideline, the author would recommend that the analyst attempt to choose the simplest objective function that incorporates their key measures of load, utilization, and response. The use of a minimum number of metrics allows the analyst to more easily anticipate how the objective function will behave over a broad range of values. However, the selection of these metrics is dependent on the specific characteristics of the installation.

3.3 Three Value Objective Function

The third objective function was defined as the un-weighted average of the Z-statistics of CPU utilization, transaction volume, and response time. The objective function was calculated:

$$\text{Obj Fun} = \frac{\text{Z-CPU} + \text{Z-Tran} + \text{Z-Resp}}{3}$$

The result of this objective function calculation is shown in **Figure 5.** The peak hour is denoted by the > and < symbols.

Hour	% CPU	Page/Sec	Avg PDT	Tran(K)	Trv Resp	
8	68	113	4.5	53	0.41	
9	77	140	4.9	137	0.65	
10	95	241	5.7	187	0.87	
11	94	237	6.3	181	0.92	
12	72	155	4.6	71	0.49	
13	86	201	5.1	172	0.80	
14	97	229	5.2	184	0.85	
>15	96	233	6.1	175	0.91<	
16	87	198	5.1	162	0.75	
STD	10.37	44.56	0.59	47.69	0.17	
MEAN	85.78	194.11	5.28	146.89	0.74	
						OBJ FUN
8	-1.71	-1.82	-1.31	-1.97	-1.89	-1.86
9	-0.85	-1.21	-0.64	-0.21	-0.51	-0.52
10	0.89	1.05	0.71	0.84	0.75	0.83
11	0.79	0.96	1.72	0.72	1.04	0.85
12	-1.33	-0.88	-1.14	-1.59	-1.43	-1.45
13	0.02	0.15	-0.30	0.53	0.35	0.30
14	1.08	0.78	-0.13	0.78	0.64	0.83
15	0.99	0.87	1.38	0.59	0.98	0.86
16	0.12	0.09	-0.30	0.32	0.06	0.17

Figure 5. Three Metric Objective Function

Using the three metric objective function identified hour 15 as the peak interval for the study. Once again, hour 14 during which the highest CPU utilization occurred was not selected by the algorithm.

4.0 Remarks

In this paper, an objective function approach for identifying peak intervals in measurement data has been discussed. While the analyst must still select the metrics and weights to be used in the definition of the objective function, it provides a structured approach to the identification of peak periods in large volumes of measurement data.

5.0 References

[1] Artis, Forecasting Computer Requirements: An Analyst's Dilemma, ECOMA 9, 1982.

[2] Deitch, Analytic Queuing Model for CICS Capacity Planning, IBM Systems Journal, Volume 21, Number 4, 1982.

[3] Simpson, Implementing an Analytic Model for CICS Systems, CMG '83.

WORKLOAD-ORIENTED PERFORMANCE ANALYSIS IN MVS

Stephen L. Samson
!Candle® Corporation

ABSTRACT

The analysis of acute performance problems in large multiprogramming operating systems such as IBM's MVS has evolved from resource-based views to those centered on the workload in trouble. A recently-implemented technique builds on the workload view, looking back at other workloads in the system to pinpoint sources of resource contention. The performance analyst as detective is now able to depend on eyewitness testimony rather than on circumstantial evidence. In this paper, we will examine the various techniques and their use when responding to performance problems.

INTRODUCTION

Acute performance problems in MVS systems have been investigated from numerous points of view. If we had perfect insight into workloads and their relationship to one another and to the operating system, we'd see that when a workload fails to meet its service objectives its access to resources is usually impeded by some other workload. Rarely is it true that the programs supporting key workloads are so flawed in design that they continually incur self-contention, nor are they so hungry for resources that they simply cannot get enough CPU, storage, or data to meet their service targets. Although many counterexamples do exist, a single workload usually runs well all alone.

Unanticipated and severe performance problems on machines with well-understood complex workloads can hardly ever be capacity problems. Unfortunately, we often look first at resource data also used by capacity planners just because it is available. Thus as we examine the state of the art in analyzing, and devising solutions for, acute performance problems, we find that we are almost always looking at the wrong data. If we want to know why "CICS has gone silly," do we look at the system to discover which other workload has the CPU when CICS wants it? Of course not! We might be able to discover which workload is receiving the most CPU service in the current time frame and, in the interest of doing "something...anything!" we cancel a poor innocent batch job running at such a low dispatching priority it couldn't possibly be in contention with CICS. At the same time, we miss the real culprit: JES2 is busy dumping a 40 megabyte file through a channel-to-channel adapter to another system. We cleverly offloaded printing, only to cause JES to dump data into an efficient bottomless pit. (All the CPU time attributable to JES did not show up because it builds a very efficient, tight loop around EXCPs. The path through I/O completion code is much longer than the path through JES.)

In this paper and its companion presentation, we'll examine the awakening of an installation to the possibility of MVS performance problems and consider several approaches to solving them. Of necessity, the last solution presented will introduce a new approach now implemented exclusively in a product of Candle Corporation. We mention this fact at the outset to prepare the reader for what may appear to be the presentation of product information. This is not the purpose of the examples in the presentation; they are there only as a concrete demonstration of the value of directly observed workload contention. Product identification has been removed from the examples.

This paper roughly parallels the presentation. Cross-references are not included in the text, but the visuals are attached as an appendix.

EARLY STAGES OF A NEW MVS SYSTEM

Bringing up a new system

As a new MVS system is installed, say as a twin to an existing system to deal with workload growth,

©Copyright Candle Corporation 1987. Permission is granted to The Computer Measurement Group, Inc. to publish this paper for the exclusive use of its members. All other rights reserved.

performance is not a consideration at first. There is so much work for the system programmer[1] to do:

- Ensure that all of the operating system is installed at a consistent level. Our definition of the operating system might encompass BCP, DFP, and JES.
- Verify the maintenance level of the operating system and add selected fixes known to be needed.
- Add the next layer of vendor-supplied "system" code to the operating system core. At this point, there is not yet a meaningful workload to exercise the system.
- Verify consistent maintenance levels and rectify any detected conflicts.
- Verify IPL and basic function.
- Add vendor-supplied subsystem packages such as CICS.
- Verify consistent maintenance levels and rectify any detected conflicts.
- Verify IPL and basic function.
- Add exits and local modifications to the operating environment.
- Verify consistent maintenance levels and rectify any detected conflicts.
- Verify IPL and basic function.
- Add extensions to subsystems (other vendors, home-grown).
- Verify IPL and basic function.
- Install or re-install application packages.
- Verify IPL and basic function.
- Prepare customized PARMLIB and other parameter files to support local standards.
- Verify IPL and basic function.
- Apply a test workload and verify functional soundness.
- Gradually bring on more comprehensive functional test workloads.
- Apply a "stress test" workload (usually synthetic) and verify that performance is satisfactory at or beyond planned capacity.
- Turn the system over to production.

Even the penultimate step in this long process, the "stress test," is often incapable of detecting incipient performance problems. The stress workload is difficult to structure to represent the dynamic interactions of real workloads; it is usually deemed adequate if it can verify gross capacity. Thus the real workload is applied to the system with little more than a warm feeling that function has been verified, and that the hardware configuration *should* be able to do the job. It should hardly be a surprise, then, when performance anomalies develop.

Initial production operation

Few new systems come into production fully loaded, so it is likely that few performance problems will appear early in a system's life. Following the analysis for CPU queuing employed by Mr. Siebo Friesenborg of IBM, as well as by my colleague at Candle, Mr. Steven Hackenberg, we note that the probability of some CPU queuing increases rapidly for a workload when the percentage of CPU utilization at higher dispatching priorities exceeds 50 percent. Suppose that CICS is the "loved one" (in IBM's vernacular) of our system and that it takes up 20 percent of the CPU capacity. If JES, TSO, and system activities at higher priorities than CICS take up 35 percent, CICS is at the point of beginning to sustain measurable CPU delay, reflected internally as "wait on dispatch." Note that the CPU may be only 55 percent busy.

If we now add more workload to this system, we will affect CICS's performance. If the added workload is low-priority batch, the only activity in contention with CICS for the CPU is the increase in JES consequent to batch SPOOL activity, as well as some increase in IOS due to the batch jobs' I/O. We might load the system at close to 100 percent CPU busy without appreciably increasing CICS's CPU delay.

If on the other hand we choose poorly, we can hurt CICS significantly with far less of an increase in gross utilization. If TSO first period is higher in dispatching priority than CICS, an additional 20 percent utilization due to trivial TSO will more than double the CICS CPU delay, even though the CPU is still "only" 75 percent busy. Figure 1 shows that for the case cited, elongation due to CPU contention increases from about 80 percent to about 220 percent. A CICS transaction requiring 200 milliseconds of CPU time will wait about 160 milliseconds for CPU service before the system load is increased. After the increase of TSO usage, the delay is about 440 milliseconds.

This disproportionate increase in delay due to an increase in utilization applies to other resources as well. Paging delay is insignificant until the available frame queue is depleted to the point that page stealing will take away vital frames. I/O delay is no problem until a buildup of device utilization starts to cause device queuing, or a rise in path utilization leads to measurable RPS miss. (At that point, the increased device utilization consequent to RPS miss also tends to produce device queuing delay.)

---

[1] We recognize that "performance analyst" and "system programmer" usually refer to the same person, and that only rarely is that person known as "capacity planner" as well.

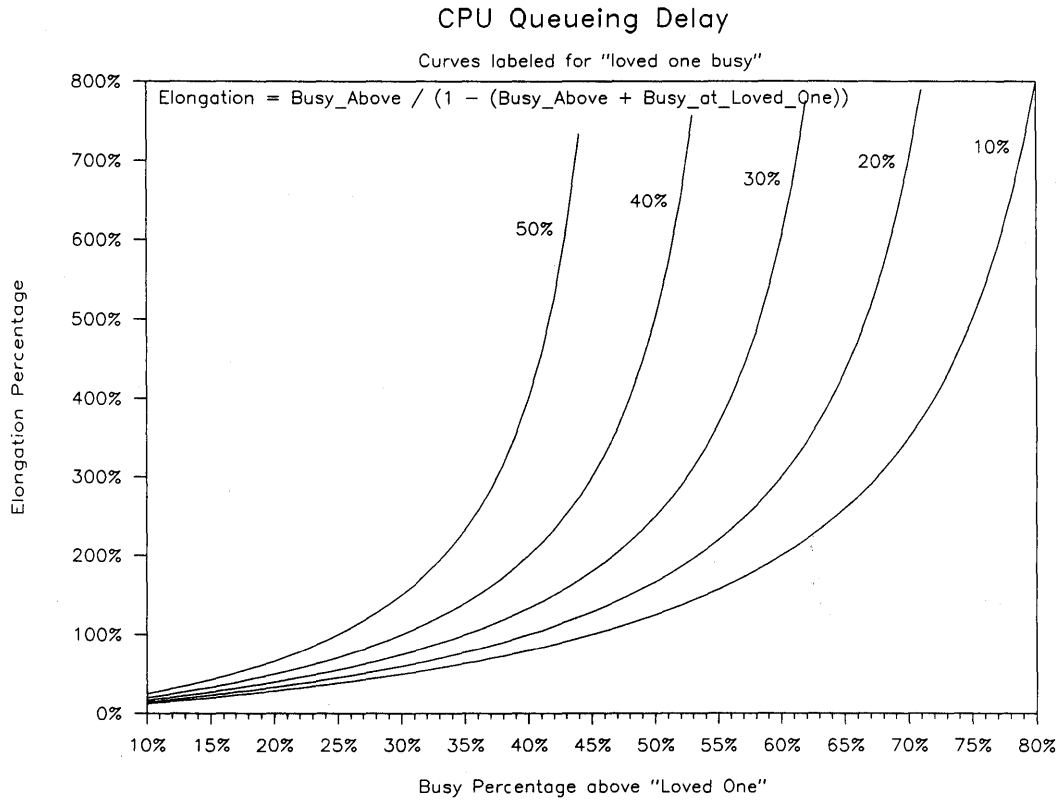


Figure 1. CPU delay family of curves

As we create models for these phenomena and analyze their outputs, we often make unwarranted assumptions of randomness. We move deftly from a discussion of CPU utilizations to one of expected CPU queuing delay, transforming utilizations into probabilities as if they were equivalent. In many cases, they are. We may assume reasonably that TSO and CICS are independent. But what of JES or a file server to CICS? If demand at dispatching priority above that of the "loved one" is synchronous with activity in the "loved one," random models are far too optimistic, and delay may be virtually certain, even at modest CPU utilization.

Initial overt problems

We might picture the system happily running with no perceived performance problems while the subclinical conditions grow, waiting to burst forth as full-fledged incidents. We define an *incident* as a detected failure to meet some service objective. With luck, the data center staff may be the first to detect an incident. This rarely is the case. The system may, after a few months of production operation, have no comfortable margin of CPU or storage or I/O capacity, and the first routine fluctuation in demand that goes beyond a critical threshold will trigger an incident.

Loss of innocence

Some time after the support staff has become convinced that they can do no wrong, the first messy incident will take place. The MIS Director may be enamoured of his Resource Scheduling System and its flashy color displays. He is demonstrating it for the CFO's budget planning meeting. Just as he prepares to make his climactic point, a batch job is being freed from allocation recovery. The operator names the device containing the Resource Scheduling System, because it's private, non-production, not familiar, and it has four hundred cylinders free. (A quad of scratch volumes is down for concurrent preventive maintenance, causing the allocation recovery event.) The first step of the batch job restores an archived data set. The tape is mounted just as the MIS Director presses **ENTER**. The very efficient restore program creates ***long*** channel programs. Even with priority I/O queuing, the ten I/Os needed serially by the MIS Director are interspersed with hundreds of I/Os for the restore....

In the aftermath of the "demo debacle," management reacts as it must; committees are formed, recommendations are made, and tools are acquired (or just put in use if they hadn't been.) Other less acute incidents are evaluated, and the realization grows that service is not all that it could be. Based on the input of

hardware vendors and out-of-town experts, a plan is devised and put into effect.

AN INITIAL PLAN—ROTS

In the absence of any solid information about contention or degradation (execution state analysis), it is difficult to see how an initial plan could be based on anything but rules of thumb ("ROTs"). The theory of this approach to performance management is simple. If you measure what you can when the system is running well and repeat the measurement over time, accumulating a data base of measurements, you can infer the normal resource utilization pattern of the system, and even infer a set of normal operational limits. You confirm that, when the system is running poorly, one or more of these measurable values is outside of the limits.

You may also choose to adopt ROTs reflecting received wisdom, straight from the out-of-town expert.

Once you have demonstrated the correlation between poor performance and operation outside of some limits, you can begin establishing a series of thresholds that may be automatically applied against the system by a product capable of performing exception analysis, such as Candle's OMEGAMON®/MVS. Whenever a threshold is tripped, an exception message is presented to the operator, and corrective action can be initiated.

Taking action with ROTs-based performance management

Using ROTs is a utilization-based approach to performance management. An exception in such a methodology is an observed resource utilization level that violates a threshold. Once an exception is detected, corrective action is based on the same kind of information. The goal is to change the utilization level of the resource and return to the normal operational range. So far, so good.

The difficulty in ROTs-based performance management appears at this point. How does one determine the appropriate action? There are numerous sources of advice on how to proceed in this situation, but it is too easy to fall into the trap of indiscriminate load shedding. If a subsystem workload like CICS or IMS is the subject of response time complaints and an exception for "excessive CPU utilization" is displayed, the natural reaction is to cancel whichever batch job is using the most CPU resource at the time. We know that cancelling a job that will be replaced by the next job on the queue will not work in the long run, but isn't it satisfying to do *something?*

More considered approaches are based on longer-term data and may lead to tuning atrocities such as imposing restrictive constraints on batch domains, activating response time objectives (RTO) on TSO first period, adding a delay domain in TSO, and cutting back to a deliberately inadequate number of initiators. All these actions are consequences of utilization-based performance management and either second-guess or contradict MVS's workload management mechanisms.

The pessimistic and inherently inflexible state of mind engendered by a dependence on ROTs also may lead to ill-considered hardware procurement. Many system configurations become distorted and unbalanced by the addition of unnecessary fast paging devices or even CPU upgrades in response to a perceived resource shortage, solely the result of exceeding an arbitrary threshold and concluding that a resource is exhausted.

Drawbacks of ROTs-based performance management

It might appear that the type of actions that ROTs-based performance management leads to are enough of a reason to reject the approach. Several additional considerations add greater force to the rejection:

- Resource exceptions have little or no relevance to service levels.
- Service exceptions are poorly correlated with resource measurements.
- ROTs are based of necessity on wisdom of the past.
  - Obsolete models and legends
  - Obsolete workloads and structures
  - Major lag when systems change
- Cause and effect are confused
- ROTs attempt to level special cases

A LOGICAL TUNING APPROACH

Several years ago, as MVS production systems got to be of crucial importance to the enterprises in which they were installed, more effective performance management disciplines became necessary and started to appear. These new methods were built on developments that occurred in that time frame:

- Service level targets, agreements, and contracts appeared and matured.
- Measurement tools, supporting programs, and data-base systems capable of tracking the performance data needed to administer service-level agreements came into widespread use.
- Other measurement tools capable of revealing a detailed breakdown of elapsed time or response

time became available. With such tools, the detection of a service anomaly can lead to rapid discovery of specific bottlenecks.

- The basic mechanisms of MVS were refined and improved, and many impediments to system growth and flexible workload management were overcome. This process continues with the MVS/XA releases that have appeared recently.

The net effect of these developments made a more logical approach to MVS performance management possible. The new methodology was superior to a purely utilization-based approach. The stepwise approach, called the "logical tuning approach" by Candle Corporation, may be executed in real time for acute performance problems, or after the fact for problems that are detected over a longer time interval. It consists of the following elements:

1. Is there a service-level problem? If not, return to routine monitoring.

2. Determine the nature of the bottleneck responsible for the service-level problem. ("Where does it hurt?") Use Degradation Analysis (also known as execution-state analysis or response-time-component analysis) to rank the execution states of the workload failing to meet its target.

3. Select the most prevalent tunable execution state (*e. g.* "waiting for CPU") and identify the cause of the contention or resource exhaustion. The cause will usually be another workload or, perhaps, an MVS parameter that can be adjusted so that the affected workload will be more likely to obtain its required resources with reduced contention. Execution state analysis will also reveal competing workloads in many cases, such as which other performance group is using a device needed by the "loved one."

4. Take appropriate action. Examples include setting a new IPS to improve dispatching priority or storage isolation, swapping out or requeuing and cancelling a job causing I/O contention, or PAGEADDing an extra page data set to cope with a temporary increase in paging activity.

5. Finally, return to routine monitoring of service targets.

Note that "routine monitoring" with degradation-oriented tools can lead to an appreciation of the normal profile of execution states for each key workload. Tracking such profiles over time with emphasis on delay states can provide valuable early warning of impending overload incidents. Such tracking is a valuable adjunct to conventional capacity projection methods.

Pros and cons of the logical tuning approach

The logical tuning (or degradation-based) approach is simpler to use and more precise than a resource-based approach leaning on resource ROTs. The focus is correctly on the workload in trouble, and initial investigation leads rapidly to the most promising resource for adjustment of demand or tuning effort. If the relief chosen is a tuning effort with respect to the workload not meeting its service targets, this method can lead to appropriate corrective action quickly and accurately.

The degradation-based approach is less successful when tuning of the ailing workload is not an effective response. In the case of acute contention for a resource by competing workloads of greater or equal importance than the afflicted one, the question that needs to be answered is not "Where do I hurt?" so much as "Who's hurting me?" In such a case, all that the degradation-based approach can do is tell who else is using the scarce resource, and we are led again into the error of confusing concurrent utilization with contention.

WORKLOAD-ORIENTED PERFORMANCE MANAGEMENT

What's needed to solve the acute contention problem is a way of determining, from the viewpoint of a workload of interest, which other ***workload*** is keeping "me" from getting the resources "I" need. This is ***not*** the same as finding out who else is using the resource in the same time frame I am. If we imagine a sampling monitor, the basic cycle of operation would not be to classify execution states of address spaces, but to look for address spaces that require service but are not currently receiving it. Such address spaces may be on the dispatcher's "ready" queue, waiting for page fault resolution, waiting for swap-in, queued on a UCB for a busy device, and so forth. Once such address spaces are identified, the analysis goes on to determine which other address space is currently using the needed resource.

Impact analysis

Candle Corporation has recently introduced the first product embodying this methodology, called *Impact Analysis*. It is in a CICS product, but the methodology is certain to appear in other environments as well. Once a failure to meet service objectives is known or suspected, a few minutes worth of Impact Analysis data collection can reveal which other workload is preventing the workload of interest, CICS in this case, from meeting its service objectives.

Contrast with older methods

Impact Analysis (IA) is quick and efficient. If the dispatching priority of the collector is high enough, occurrences of disproportionate contention such as those caused by synchronous invocation of server code can be revealed. As in Degradation Analysis, the data shown by Impact Analysis needs to be consulted in the context of a service level exception if conclusions drawn about indicated contention are to generate action. An impact display or impact profile for a workload behaving normally may show contention, even high values of contention. Such a display simply reflects the fact that queues exist in MVS and that some delay is the price for the considerable benefits of multiprogramming and shared use of system resources.

When synchronous contention at higher dispatching priority is seen to be a significant cause of delay, we become able to evaluate hardware configuration and software architecture choices more effectively. The effect of such contention may be minimized better in a multiprocessor than in a uniprocessor, and in a muli-task structure rather than in a serialized software architecture.

Properly drawn service-level agreements acknowledge the existence of delay and do not commit the service provider to strive for theoretical minimum response times, but for attainable response times that meet the business needs of the users.

CONCLUSION

There has been an evolution in methods to deal with acute performance problems in MVS. The earliest approach based on measured data rather than on intuition used resource data only, and exception detection was based on violation of resource-based rules of thumb (ROTs). Corrective action (if possible in real time at all) was usually based on ranking the users of the constrained resource and taking some drastic action against the most prominent user not known to be "important."

The second stage of the evolution became posssible with tools performing Degradation Analysis (Execution State Analysis) and with the rise of Service-Level Agreements. In the degradation-based methodology, exception detection is based on failure to achieve service level targets, and corrective action is based on information gathered over time by a degradation analysis collector. This methodology focuses more rapidly on specific problem causes than did the earlier one, but corrective action is still largely resource-based. Resource-based corrective action is usually effective if it is a positive action to improve access of the workload in trouble to the resources it needs. Changing storage isolation or dispatching priority are examples of such effective positive actions. Negative actions against apparently competing workloads, however, are almost as likely to be misdirected or ineffectual in this methodology as they are in a purely resource-based system.

The most recent stage in the evolution is marked by the introduction of Impact Analysis. A problem-solving methodology based on Impact Analysis continues to be dependent on service level analysis for detecting exceptions. Impact Analysis for problem source analysis and for response selection captures the essence of "speed of use," leading rapidly both to correct identification of contending workloads and to decisive corrective action.

ON THE OPTIMIZATION OF MEMORY IN LARGE INTERACTIVE ENVIRONMENTS

Anthony G. Mungal.

ABSTRACT

Large interactive environments can put a significant demand on the memory resources of a processor. Memory is an expensive finite resource; proper exploitation of which can contribute to enhanced workloads and overall greater throughput. In environments where cost is of little concern, it is advantageous to configure the processor with its full complement of memory. That necessarily implies that most, if not all, workloads that compose the multi-programming level will suffer fewer page faults, and hence, experience their maximum throughput (to the constraints of the CPU itself and the I/O subsystem).

In other environments, where the economic benefits of additional memory are not as easily felt, controlling the amount of logical swap incurred by the interactive system, can result in an effective tuning of the memory resources, such that, the productivity associated with low and consistent response times remain at an economical advantage.

This paper discusses a technique whereby main memory is "tuned" effectively to achieve the best compromise between job throughput, response times and cost; it employs a technique of off-loading a certain amount of logical swapping, allowing more "free" memory to exist. Indicators of memory contention are examined and interpreted in order to evaluate the best cost/performance alternative. This analysis (which utilizes actual customer data) will answer the question : How much memory is optimal?

The existence of more free memory naturally results in a lowering of the page fault rate, and thus, a more productive system can be realized.

INTRODUCTION

The trend of data processing environments is related to many concurrent evolutions of various pieces of hardware. One can look at the trends of at least the following pieces of hardware: the processor, memory, and the I/O subsystems. While those trends continue, the capacity planner struggles with the decisions and justifications of at least the following: the size and capacity of the current and/or future processor(s), the size of main memory, the appropriate number of channels, the number of users, the working set size of the user, and of course, the associated response times. In trying to evaluate the usage of memory within these sprawling interactive environments, one must look at the various interactions of many variables.

A methodology exists for optimizing main memory through the careful selection, interpretation and subsequent monitoring of certain memory indicators; those indicators are discussed within the context of an MVS/TSO environment. For TSO environments, logical swapping is commonly utilized to maintain low, consistent response times. That may be to the detriment of other workloads within that environment.

This paper discusses the necessary indicators of memory contention, and suggests proven alternative methods for optimizing the useage of memory. Much reference is made to an assortment of large environments which indulged in the same or similar analyses; data from those environments is included. The latter portion of this paper deals with the cost/performance or economic issues surrounding the deployment of memory on similar processors, and hence, forms the basis for evaluating the effectiveness of the capacity planning decisions involved.

WORKING SET DEFINITION

The definition of the working set as it applies to a job or interactive user has received increased attention over the years. Denning [11] proposed its concept many years ago. Garay et al [7] formalizes a rather useful definition of the working set as it applies to an interactive user; it is summarized below:

Let the set of addresses be

$X = \{x1, x2, \ldots, xn\}$. Denote a

reference string of length k as a sequence r, such that

$r = \{r1, r2, \ldots, rk\}$ and each member of r

belongs to X. The duration of reference is denoted T, a positive integer such that

$T = t1 - t2$ for all $t1 > t2$.

The working set during T thus becomes a reference string W(T), such that

$$W(T) = \{r1(t), r2(t), \ldots, rk(t)\}$$

and $t \in [t1, t2]$, thus providing the set of all referenced pages during the interval t. Now W(i) can denote the working set of the ith user during execution. The average working set size of all users becomes simply,

$AW = W(1) + W(2) + \ldots + W(n)$ or

$$\sum_{i}^{n} W(i) \qquad n > i$$

The cardinality of the working set (working set size) can be denoted

$$s(i) = \lceil W(i) \rceil$$

s(i) is further subdivided into swap sets (each of 12 pages), forming disjoint subsets of W(i), so that swapping can handle the individual subsets

$$S1(i), S2(i), \ldots, Sm(i)$$

where m is the number of swap sets. Note the potential for Sm(i) i.e. the last swap set, to be less than 12 pages.

The above terminology will prove useful for certain situations later.

TSO ISSUES

THE TSO USER

In looking specifically at the TSO environment, it is useful to trend the address space size of the user as it evolved through various releases of MVS. For example, observations indicate that the average TSO user increased in size by at least fifty percent in going from MVS/SP to MVS/XA. Also, the XA operating system itself allows the user to exploit a much larger address space (thirty one bit addressing), thereby allowing him a two gigabyte address space as an operating domain. Those two factors, taken by themselves or together, clearly has some significant impact on the paging and swapping loads observed within some systems.

The typical TSO user has had an increase in his working set size, which in turn reflects the potential for an increased amount of demand paging (page faults) in order to achieve transaction completion. The increase in demand paging per transaction per user has resulted in dramatically higher page rates for the overall paging subsystem; this has resulted in the adaptation of very robust paging and swapping subsystems. Clearly, the need for page and swap subsystems that can handle high page rates and provide low, consistent response times depends on the size of the processor, the number of users, the number of transactions, and of course, the amount of real and expanded storage (memory).

TSO RESPONSE COMPONENT ANALYSIS

An examination of the components of TSO response times reveals various entities such as: I/O time, CPU time, page-in time, and swap-in time. Beretvas [2] details some calculations that enable a post facto component analysis for TSO. In the majority of environments studied, the major contributor to degraded response times was swap-in delay. Swap-in delay is the time taken to bring a user from auxiliary storage (page or swap datasets) to main storage; it can also be measured/calculated for users brought in from expanded storage (in the microsecond range). Swap-in delay is easily calculated from Resource Management Facility (RMF) data, and is detailed below:

swap-in delay = (average absorption rate
average transaction service rate) *
average time of ended transaction

The correction term that may be applied to the above swap-in delay value for situations that employ a large amount of logical swapping is a multiplier of the form 1/(1 - %logical swap).

It should be noted that the TSO response time parameter (RTO) can be used to limit the service given to TSO. This is accomplished by the SRM delaying the TSO transaction after the command or other terminal input but before swap in occurs; this can impact swap-in delay values and the user is urged to discount swap-in delay calculations accordingly. A good discussion of RTO can be found in [12].

SWAP-IN DELAY AND PATH CONCURRENCY

It is desirable to maintain low swap-in delay in order to experience low, consistent response times. Many swap subsystems (especially those on rotating disk) will not ascribe to the property of low, consistent service times under load, and hence, are unable to guarantee low and consistent response times; the exception to this is the solid state disk. This solid state subsystem is discussed by Moore [12], Cheung [5], and Mungal [8]. In fact, due to the inherent sensitivity of rotating disk to I/O load, rotating disk will be a dissappointment when one needs it the most (high quantity of I/O needs to be done).

As the TSO user increases in size (region), the greater the probability that the working set size s(i) will increase, thereby leading to a larger value for "m", the number of swap sets. The swap operation becomes bigger, thereby requiring more concurrency denoted "c". That concurrency is needed along swap paths, controllers, and actuators, and can be denoted "m/c"; for DASD swap subsystems, it is optimal for c to be at least equal to m; for solid state swap subsystems, it is often sufficient for c to be not less than two, and not exceeding m/2 (i.e. $2 \leqslant c \leqslant m/2$). Of course, regardless of the type of swap subsystem, swapping becomes optimized for values of c equal to m (c=m). The major benefit of maximum concurrency is the reduction of swap-in delay.

SWAP-IN DELAY AND RESPONSE TIME

Figure 1 illustrates the relationship of swap-in delay with TSO response times as observed on a 3084Q processor, configured with 64 megabytes of main memory. The erratic nature of TSO response time is clearly observable, suggesting that for various times during the shift, users are subjected to erratic response times. Also, a clear correlation between consistently low response time and swap-in delay less than 25 milliseconds is observable. The erratic response time can be easily corrected by employing a more robust swap subsystem. It should be pointed out that any swap subsystem has its capacity (in terms of I/O load); as I/O loads fluctuate close to or even exceeding that maximum load even momentarily, one can expect that response times will degrade. Hence, it is essential to ensure that such subsystems are carefully configured to accommodate expected peak I/O loads and even some unexpected ones (i.e. reserve capacity).

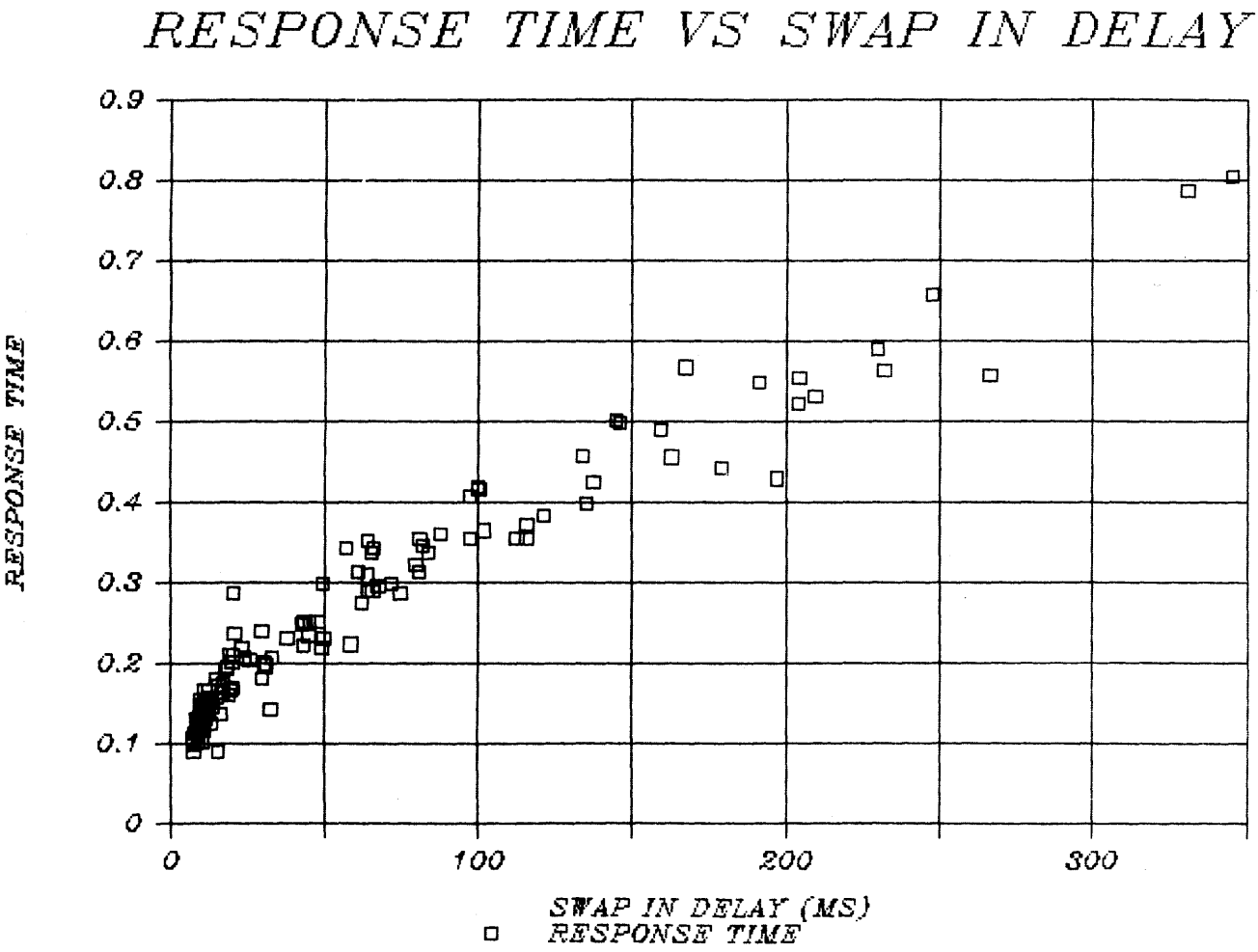


FIGURE 1 - RESPONSE TIME VS SWAP-IN DELAY (CUSTOMER DATA)

MEMORY TUNING ISSUES

LOGICAL SWAPPING

The effective use of logical swapping is a means of lowering the quantity of physical swapping while, at the same time, providing low and consistent response times. Logical swapping is a process whereby a TSO user is kept in main storage until he is again ready to execute. The logical swap process is dependent on the logical swap think time, which in turn is governed by the average system wide unreferenced interval count (UIC). Logical swapping may also be impacted by assigning certain values to the low and high thresholds of the available frame queue; the parameter, LSCTAFQ is discussed in [12].

While the overall benefits of logical swapping remain without question, the disadvantages of it are oftentimes somewhat elusive. The cost is simple: inactive users are guaranteed acceptable response times, sometimes at the expense of the active users.

UNREFERENCED INTERVAL COUNT (UIC)

If the amount of available real memory within the processor remains high, one can and should indulge in logical swapping; this should only be done in conjunction with low demand page rates. On systems with a high amount of logical swapping and high demand page rates, something more is needed; an examination of the unreferenced interval count (UIC), and the available frame queue (AFQ) is necessary

The UIC values for any given system is an indicator of memory busyness; it is typically a high value (max of 255) on systems that exhibit little or no contention for memory. A low system wide UIC (minimum 0) is a clear indicator of high memory contention. The use of logical swapping can be interpolated amongst the values for UIC and the available frame queue.

AVAILABLE FRAME QUEUE (AFQ)

The available frame queue AFQ is a reported statistic within RMF and is another indicator of memory contention. As a matter of fact, the depletion of the AFQ results in supervisor state activities which involve page stealing. It is desirable to minimize page stealing at all costs. In all the cases studied, the obvious relationship between a low or even depleted AFQ and high page-in rate for the system was observed.

STORAGE ISOLATION

Storage isolation is a facility available to the performance analyst that permits the protection of an application's working set and/or common area. The critical working set is determined and the low and high paging threshold values are set; the system now attempts to maintain the working set as defined within those thresholds. If however, memory contention sets in (low UIC, depleted AFQ etc.), the system will eventually steal pages from the protected applications.

The overall recommendation of storage isolation evidenced in both Pierce [4] and [13] is to suggest that storage isolation values (if necessary) "be selected with care and monitored closely".

In at least two of the cases studied, storage isolation was defined for "critical" applications since they shared memory with TSO and others; by reducing logical swapping in these environments in conjunction with the removal of storage isolation, more "free" memory was allowed to exist , overall system response times improved and further storage isolation became unnecessary.

MEMORY BUYBACK

Upon careful evaluation of certain large MVS systems (3081, 3084, and 3090), it was observed that a threshold amount of logical swapping was beneficial; the elimination of some (the majority) of it resulted in a better economic solution to response times and overall system throughput.

Decreasing logical swapping is achieved by lowering the system think time; this is accomplished by setting/resetting the upper and lower bounds of the LSCTMTE parameter of MVS. The effect of that (e.g. LSCTMTE=(5,10)) would suggest to MVS that users with a think time of 5 seconds or less should be logically swapped, and users logically swapped for more than 10 seconds are to be physically swapped; such a setting for think time may see logical swapping decrease from the high ninety percent range to the low twenty percent range. That resulted in a significant amount of real storage being freed, thus leading to a dramatic increase in the size of the available frame queue. The resulting increased physical swap rates can be handled easily by an appropriate number of disk actuators or a pre-determined amount of solid state disk.

The rationale for the above action is essentially predicated on two main reasons. In the first place, users that take a long time to think of their next action should not tie up a valuable resource such as memory; they should be removed in favour of the "fast thinkers". Secondly, main memory availability determines the quantity of logical swapping within the system; it is a fluctuating quantity, which influences the systemwide UIC values and the unused frame count. Hence, one must judiciously oversee the overall committment of main memory to logical swapping, since periodic overcommitment can and will result in storage contention (low UIC values), and even, page stealing when the unused frame queue becomes depleted.

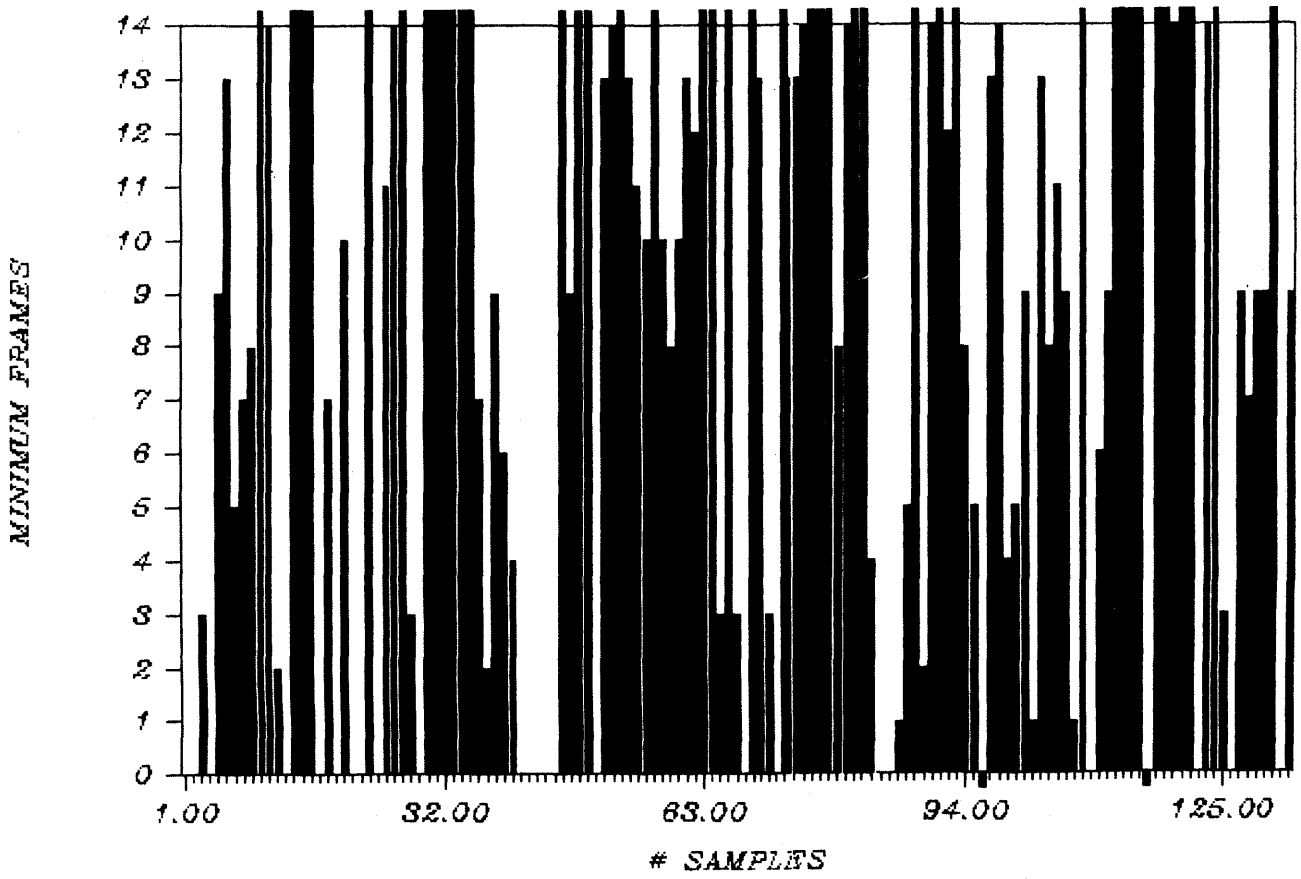


FIGURE 2 - MINIMUM FRAMES DURING PEAK SHIFTS (CUSTOMER DATA)

Figure 2 details the fluctuating values for the available frame queue on a 3084 processor with 64 megabytes of main memory, while the quantity of logical swaps was averaging about 85 percent. That situation was easily rectified by lowering the system think times, which resulted in a lowering of logical swapping, an overall reduction of the page-in rates, an increase in physical swapping, and an increase in the UIC values for unreferenced frames.

The amount of memory freed (buyback) is evidenced in Figure 3. In at least one instance, the "buyback" of existing real storage served to decrease the demand page rates significantly. Upon further analysis, it was also observed that other workloads benefited from the decrease of the demand page rates; a primary beneficiary was IMS. On other systems, one may expect the primary beneficiaries would be any shared address space. The benefit conferred upon IMS (a significant decrease of demand paging) is illustrated in Figure 4.

In achieving the aforementioned results, the available frame queue increased dramatically and system wide UIC values were lowered (although not to the point that would suggest increased memory contention).

EXPANDED STORAGE

The 3090 series processors provide the option of an additional amount of memory that can be used as a repository of page and swap pages; this is known as expanded storage. Transfers of pages from expanded to main and vice-versa occurs in units of 4K. The expanded storage is under the control of the real storage manager, and it is not user addressable. This level of storage is not as costly as main memory, but, to date, remains more expensive than any currently available external page/swap subsystem.

Expanded storage holds pages that were infrequently used in main memory, as well as logically swapped TSO users. This can be viewed as yet another option of tuning main memory since pages that are least recently used (LRU) are LRU'ed out to expanded storage by the real storage manager.

Various experiences with expanded storage suggest the following: reduction of paging and its associated I/O, hence better response time; little or no additional throughput. In fact there is mounting evidence to suggest that due to the synchroneity of the processor during page reference to expanded storage, the

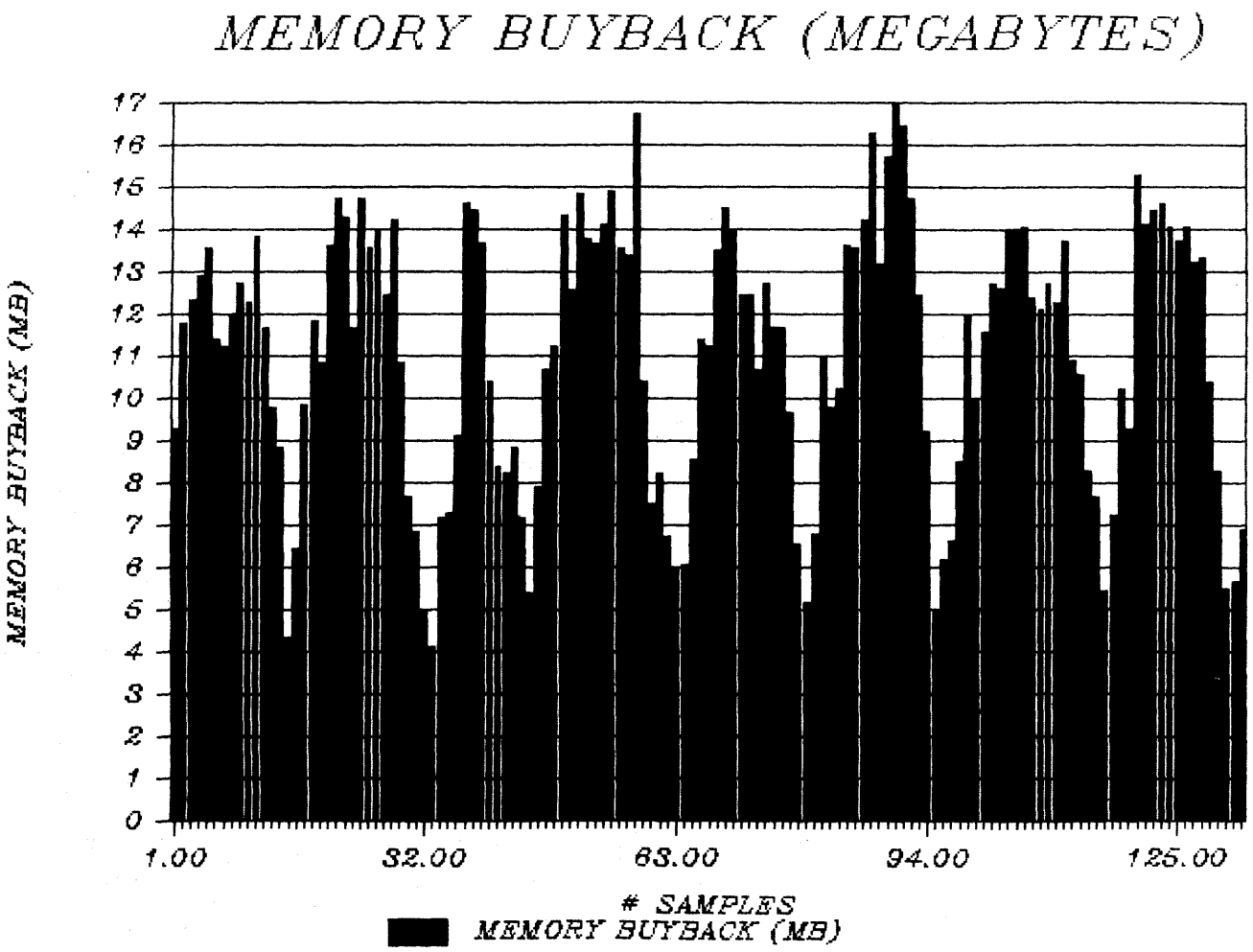


FIGURE 3 - MEMORY USED (LOGICAL SWAPPING ONLY) DURING PEAK SHIFTS (CUSTOMER DATA)

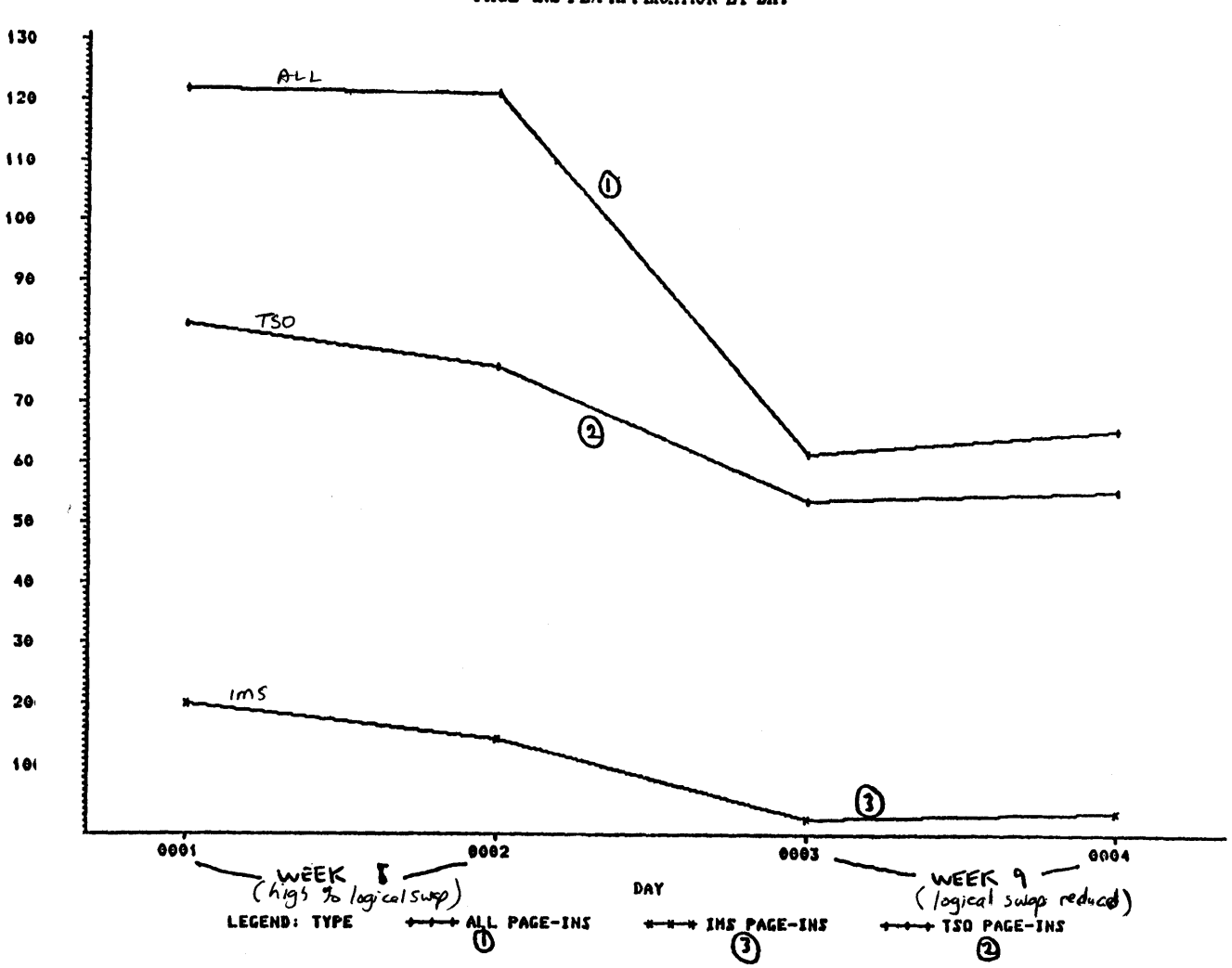


FIGURE 4 - AVERAGE DAILY PAGE-INS PER SECOND DURING PRIME SHIFT - BY APPLICATION (CUSTOMER DATA)

quantity of traffic to and from expanded storage imposes some very real constraints on processor throughput.

Other vendor's implementation of expanded storage involve the logical partitioning of the available real memory. The experiences thus far, suggest that this approach is not similarly constrained as the 3090.

SHARED vs UNSHARED ADDRESS SPACES

A shared address space such as IMS or CICS allows multiple users to execute within it. An unshared address space is one that facilitates only one user to execute within it; this is typified by TSO.

The fundamental difference between the shared address space and the unshared one, in terms of their respective memory requirements is the following: when a page fault is suffered by a shared address space, all users within it wait until it is resolved, even though it may have been caused by one distinctive reference on the behalf of one user. When a page fault is suffered by an unshared address space, only that user waits.

In terms of the reference strings so evolved, the shared address space will tend to exhibit a more diffuse locality of reference for its reference string W(T), since it is attempting to satisfy the requirements of many users during the reference interval t. The unshared address space reference string W(t), and working set size s(i) is such that

$$s1(i), s2(i), \ldots, sm(i)$$

accounts for the locality patterns for m unique users within the address space and thus, represents m mutually disjoint locality sets. Further, the m locality sets identified, may be unequal in cardinality (unlike TSO), and would each generate their own reference string. A more diffuse locality of reference across the working set s(i), will result in a higher probability that the frame that needs to be re-referenced will be stolen prior to that re-reference and hence, a page fault will be suffered. This provides the basis for different memory considerations for such address spaces.

The TSO user, on the other hand, enjoys great consistency across reference intervals (swap-ins), and thus contributes to a smaller locality of reference.

COST/PERFORMANCE ISSUES

Throughout, the issues of determining the judicious usage of memory has been discussed, and alternatives presented. The capacity planner may still agree that recommending memory so as to reduce paging and/or swapping is the safest and most proven thing to do. However, one should consider the useful remaining lifetime of the intended processor and the cost of acquiring more real storage versus the adoption of an optimal paging/swapping subsystem, and then proceed from a cost/performance viewpoint.

Table 1 details the relative cost of the various storage alternatives along with their response time characteristics (swap-in delay, standard deviation). It is readily apparent that real storage (highest cost) will undoubtedly render the best performance in terms of low and consistent response times. Expanded storage, which is relatively cheaper than real storage may work well (almost zero swap-in delay) until the page movement to and fro increases, at which time, throughput and response times may be negatively impacted. The solid state disk affords the user a small swap-in delay (10-25 milliseconds), coupled with a flat, consistent service time at a lower cost.

The consequences of zero swap-in delay is obvious; that of 10-25 milliseconds serves to increase the TSO response time by 0.01 to 0.025 seconds -- the effect upon the user is negligible.

	SWAP-IN DELAY	STANDARD DEVIATION	RELATIVE COST
REAL STORAGE	0	LOW	4
EXPANDED STOR	~0	LOW	3
SOLID STATE	10-25	CONSISTENT	2
ROTATING DISK	30-??	ACCEPTABLE ?	1

COST: 1 (LOWEST) - 4 (HIGHEST)

TABLE 1 - SWAP-IN DELAY AND RESPONSE TIMES FOR VARIOUS DEVICES

It should be realized that the cost, configurational flexibility, performance and portability of a high speed paging/swapping subsystem will allow any environment a considerably better level of cost/performance.

CONCLUSION

Current processors already offer a myriad of main memory sizes in addition to the minimum recommended (standard) amount for that particular machine; in addition to that, some others even offer expanded memory. The decisions for the capacity planner with respect to configuring an optimal amount of memory can be as simple as "all that can be configured" or "all that can be effectively utilized".

The "better safe than sorry" attitude of some capacity planners often leave little recourse for pursuing a detailed analysis of memory useage; that forms a stark contrast against those who relentlessly search for "optimal solutions" for their problems. There is no performance or capacity problem that can't be solved by "throwing" hardware (money) at it.

Shared address spaces have a somewhat different memory requirement from non-shared address spaces such as TSO. Optimizing memory is simply a matter of interpreting the indicators of contention within the context of the existing workload; all workloads are not created equal and hence, all machines need not be configured the same!!

ACKNOWLEDGEMENT

I would like to thank my wife, Saroj for her painstaking efforts in the typing of this manuscript; also, for her patience and understanding during her many lonely hours while I prepared it. Special thanks to Paul Hill, Andy Kendal, and Dave Irwin for their valuable input from time to time.

REFERENCES

[1] Dodson, G.W. Paging Subsystems - Past, Present, and Future. In Proceedings of the 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.520-525.

[2] Beretvas, T. Analysis of Processor Storage and Paging Configurations. In Proceedings of the 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.772-801.

[3] Mullen, J.W. MVS/XA Performance and Resource Management Facility (RMF) Data Analysis. In Proceedings of the 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.803-812.

[4] Pierce, B.R. Effective Use of SRM Controls. In Proceedings of the 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.813-829.

[5] Cheung, D. Storage Strategy. In Proceedings of 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.333-336.

[6] Major, J.B. A Methodology of Processor Capacity Comparison. In Proceedings of 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.241-252.

[7] Garay, J.A., Kienzle, M.G., Tetzlaff, W.H. Preliminary Analysis of Page Reference Strings of an Interactive System. In Proceedings of 1985 CMG International Conference (Dec. Dallas, Texas). 1985, pp.94-105.

[8] Mungal, A.G. The StorageTek 4305: A TSO/IMS Perspective, Technical Report, StorageTek Canada Inc., Toronto, Canada. 1983.

[9] Moore, F. Report on the Large Systems Storage Hierarchy, Technical Report, Storage Technology Corporation, Louisville, Colorado. 1987.

[10] Bolker, E.D., Shum, A.W., Agrawal, S.C. Modeling Memory. In Proceedings of the 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.733-735.

[11] Denning, P. Working Sets: Past and Present, IEEE Transactions on Software Engineering, vol. se-6, no. 1, Jan. 1980, pp.64-84.

[12] Moore, F. Solid State Device Considerations. In Proceedings of the 1985 CMG International Conference (Dec. Dallas, Texas). 1985, pp.136-153.

[13] IBM Corporation. MVS/Extended Architecture System Programming Library: Initialization and Tuning. GC28-1149-4.

AUTHOR/SPEAKER

Anthony Mungal is currently employed within Systems Engineering with Amdahl Canada Ltd. Prior to that he spent six years with Storage Technology (StorageTek Canada Inc.) in various positions within Systems Engineering, including product specialist, SE management and Consulting SE. His background is in systems programming with emphases on performance and tuning, capacity planning, programming, and I/O subsystems. He is a graduate of the University of Toronto (majors in mathematics and computer science) and an active member of the Association for Computing Machinery (ACM) and CMG Ontario.

Estimating the Maximum Throughput of a CYBER 930 Disk Subsystem

Andrew Tron
John Sutherland

Control Data Canada Ltd.
1855 Minnesota Court, Mississauga, Ontario

ABSTRACT

A way of estimating the maximum throughput of a disk I/O subsystem is presented by way of a case study of a Control Data CYBER 930 Disk Subsystem. The method used is based upon queueing network theory. Results are presented and the feasibility of applying this method to other systems is discussed.

1. INTRODUCTION

Disk I/O subsystems play a critical role in computer system performance. Because of this, knowledge of the maximum throughput of a disk I/O system is quite important in designing computer systems. If service demands in a disk subsystem were fixed, it would be trivial to calculate maximum throughput. Unfortunately this is not the case as a significant portion of the service demand at a disk drive may represent the effect of channel contention. Channel contention is the time the disk drive waits for a control device to be free of requests from competing disk drives. The amount of contention varies with the system load: the busier the system, the greater the channel contention. Another key factor which is often not considered is the effect of software overhead in the I/O driver itself.

In this paper a method of estimating the maximum throughput of a particular disk I/O system based upon queueing network theory is presented. The system under study is a Control Data CYBER 930 Disk Subsystem. A brief description of the system and a discussion of the queueing model used are presented in Section 2. Section 3 details the method used to evaluate the model to get the maximum throughput. Results are discussed in Section 4. Generalizations of this method to other types of disk I/O systems are presented in Section 5.

2. A BRIEF DESCRIPTION OF THE CYBER 930

The CYBER 930 is a midrange computer system characterized by an extremely large virtual memory and massive I/O bandwidth. The maximum possible system I/O bandwidth is 80 megabytes per second with up to 20 megabytes per second being architecturally possible on any individual channel. The system architecture is the standard Control Data CPU/Peripheral Processor (PP) architecture with significant extensions in the I/O subsystem:

- Peripheral processors are discrete rather than "slot and barrel" (i.e. timeshared) resulting in much faster execution of PP code.
- Peripheral processor memory has been quadrupled.
- Direct memory access (DMA) capability has been added allowing data transfers directly between peripheral processors and central memory rather than requiring them to be buffered through PP memory. However the PP remains tied up for the duration of the data transfer.
- Support for the forthcoming ANSI standard Intelligent Peripheral Interface (IPI) channel with transfer rates up to 10 megabytes per second (as opposed to the 3 megabytes/second maximum for the C170 channels used on previous CYBER 180 models).
- Additional instructions designed to reduce PP overhead for I/O setup.

The net effect of these changes is a significant improvement in both I/O setup time and data transfer rates compared to previous Control Data CYBER models as well as the complete elimination of one processing stage (data transfer between PP memory and central memory).

2.1 A DESCRIPTION OF THE CYBER 930 DISK SUBSYSTEM

A CYBER 930 Disk Subsystem consists of three types of devices: peripheral processors (PPs), control modules and disk drives. The PPs are the interface between the disk subsystem and central memory. Up to eight control modules may be attached to a PP via an Intelligent Peripheral Interface (IPI) channel. The control modules provide an interface between the disk drives and the peripheral processor, and provide control signals for the disk drives. The disk drives are Control Data FSD-2's which employ rotational position sensing (RPS) to optimize channel usage.

The transfer rate of the disk - control module channel differs from that of the IPI channel which means that the control module must be buffered for speed matching. A single-path architecture is used: each disk is connected to only one control module, and each control module is connected to only one peripheral processor. A block diagram of the disk subsystem is shown in figure 1.

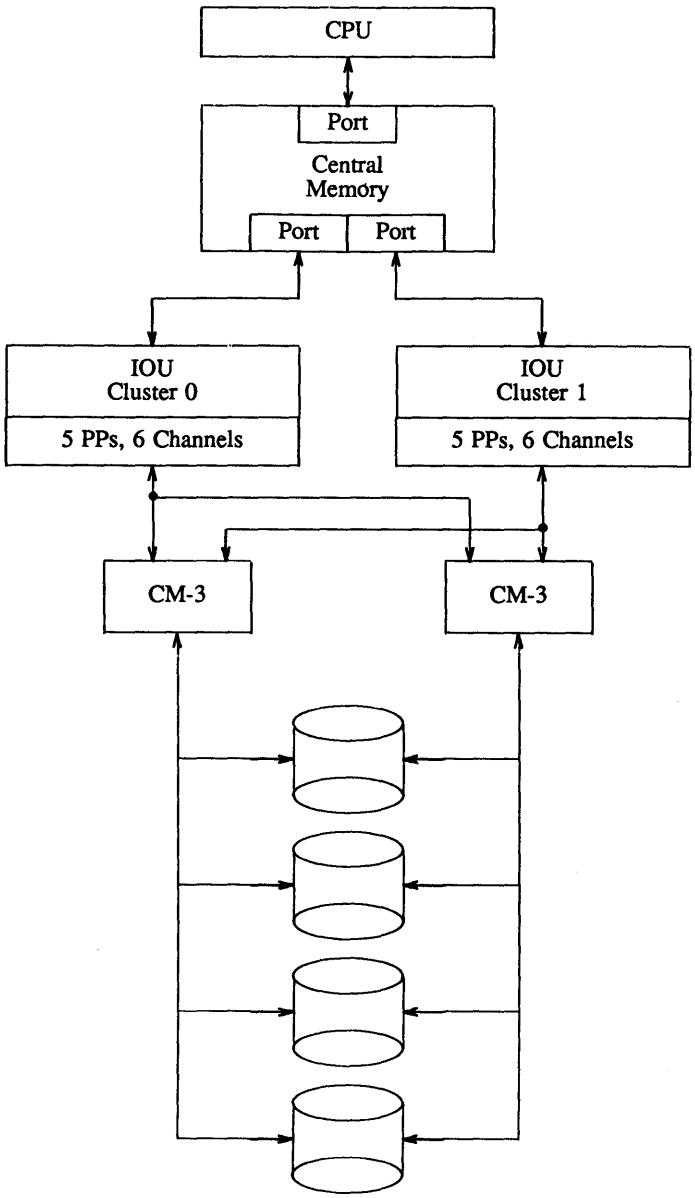


Figure 1 - Block Diagram of CYBER 930 System

Note that physically each control module is connected to two channels and each disk to two control modules. However this is a static (under software control) rather than dynamic dual access intended to provide redundancy for fault tolerance purposes. Logically the O/S software only uses one channel to each control module and drives each disk from a single control module.

Each PP is a specialized I/O processor which can transfer data between its own memory and a device on a data channel or can initiate and control a data transfer on a DMA path between central memory and the device. PPs are packaged in an I/O cluster having five PPs and six channels. A CYBER 930 can have one or two I/O clusters giving a maximum of ten PPs and twelve channels.

The control module (CM-3) is a microprocessor based disk controller with two IPI channel access paths which is capable of controlling up to eight disks. In a CYBER 930 configuration, a maximum of four disks are supported on a CM-3. The CM-3 is a buffered device having a 96K data buffer which allows it to speed match between the 10 megabytes per second transfer rate on the mainframe side and the disk transfer rate of 1.82 megabytes per second for a FSD-2 drive. The CM-3 has a software latency of about 3 milliseconds for decoding commands.

The FSD-2 disk (also known as the CDC 9836) is a nine inch sealed module drive with a formatted capacity of 414 megabytes. The disk uses a fixed block architecture with a sector size of 2048 bytes. Its nominal data transfer rate is 1.88 megabytes/second which is adjusted to 1.82 megabytes/second to allow for intersector gaps. The theoretical average seek time of the FSD-2 is 20 milliseconds.

2.2 THE MODEL OF THE DISK SUBSYSTEM

Since the control module is buffered, an I/O request can be divided into two operations: transferring data between the disk and the control module, and transferring data between the control module and central memory. This involves first queueing for the disk and then queueing for the disk drive - control module channel. In the model, this operation is represented by including each disk drive as a service centre. The wait involved in queueing for the disk drive - control module channel is incorporated into the service demand of each disk drive as channel contention. The second operation involves the control modules and the central subsystem queueing for the peripheral processor, which can be represented in the model by including the peripheral processor as a normal queueing centre. Because the control module operates in parallel with the disk and the peripheral processor, there is no need for it to be directly represented in the model.

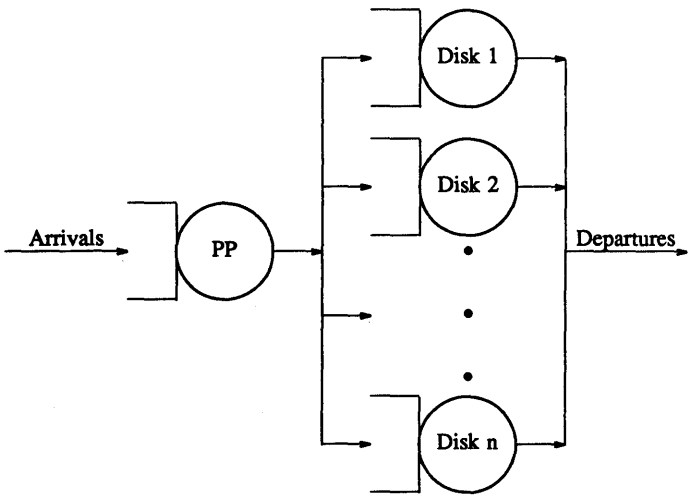


Figure 2 -- Diagram of Queueing Model

The customer population is modelled as a single open class with each transaction representing a single I/O request. When all devices are homogeneous, maximum throughput is achieved by balancing the load across all devices. Thus it is assumed that the disks have identical performance characteristics, the visit counts for all the disks are identical, and the same number of disks are attached to each control module. These assumptions simplify calculations. Furthermore, the system is modelled with only one peripheral processor. Systems with two or more peripheral processors can be subdivided and evaluated as separate systems.

3. METHOD OF SOLUTION

Maximum throughput occurs when the device with the largest service demand is saturated. Let X_{max} be the maximum throughput and let D_{max} be the largest service demand in the system. Then by the utilization law (Utilization = Throughput × Service Demand):

$$D_{max} \times X_{max} = 1$$

To find the maximum throughput, set D_{max} to each of the service demands in the model and then solve the above equation for X_{max}. The lowest value of X_{max} obtained is the maximum throughput of the system. Since the disks all have equal service demands, one need only find the service demand at the PP and at one of the disks.

3.1 MAXIMUM THROUGHPUT AT THE PERIPHERAL PROCESSOR

At the PP, the service demand consists of the time taken to transfer data between the control module and the peripheral processor plus PP overhead. The transfer rate of the IPI channel between the PP and control module is 10 megabytes/second. A rough approximation of the overhead can be obtained by assuming 600 μs per request plus 100 μs for every 2048 bytes transferred, from [2]. A more accurate, but more complex, expression for the overhead is found in Appendix A.

Let D_{pp} be the service demand at the peripheral processor and let B be the block transfer size in bytes. Then:

$$D_{pp} = \frac{B}{1.0\times10^{7}} + \frac{B\times0.0001}{2048} + 0.0006$$
$$= B \times 1.488 \times10^{-7} + 0.0006$$

Assuming that this is the bottleneck in the system, the maximum throughput then is:

$$X_{max}(pp) = \frac{1}{D_{pp}}$$
$$= \frac{1}{B \times 1.488 \times10^{-7} + 0.0006}$$

3.2 MAXIMUM THROUGHPUT AT THE DISKS

The service demand at the disks drives consists of four components: seek, latency, data transfer and channel contention. An average seek time of 14 ms is used which includes overhead of about 3 ms at the control module. This estimate of the average seek time is based upon empirical data from various benchmark runs and is less than the theoretical average of 20 ms with files evenly distributed over the disk. The reasons for this difference between observed and theoretical seek times have been discussed in the literature ([3] and [4]). The authors are aware of only one case where the theoretical average seek time has been observed on a production system [5]. The average latency period is 8.33 ms which is half the average rotational period. The average data transfer time is calculated by dividing the block size by the data transfer rate of 1.82 megabytes/second. The channel contention component cannot be parameterized as easily because it is dependent upon the system load. However channel contention time can be expressed as a closed-form function of the system throughput.

Let x be the system throughput,
s be the average seek time,
l be the average latency period,
t be the average data transfer time,
a be the average channel contention,
c be the number of controllers,
d be the number of disks per controller.

Then:

a = *rotational period* × *average number of reconnect attempts*
$= 2 \times l \times$ *average number of reconnect attempts*

In an RPS system in which the disk "sees" one level of control devices, the average number of reconnect attempts is given by [1] as:

$$\frac{U_{CM} - U_{CM}(disk)}{1 - U_{CM}}$$

where U_{CM} is the utilization of the control module, and $U_{CM}(disk)$ is the utilization of the control module due to transfers by a particular disk. U_{CM} equals xt/c, and $U_{CM}(disk)$ equals xt/cd, so that:

$$\frac{xt(d-1)}{d(c-xt)}$$

is the average number of reconnect attempts. Therefore:

$$a = \frac{2lxt(d-1)}{d(c-xt)}$$

The average service demand at the disk is then given by:

D_{disk} = *Visit Count* × $(s+l+t+a)$

$$= \frac{(s+l+t+a)}{(c \times d)}$$

since it was assumed that the system load is balanced equally across all $c \times d$ disks.

Assuming that the bottleneck occurs at the disk, the maximum throughput is found by solving the equation

$$D_{disk} \times X = 1$$

for X. This equation can be expressed as a quadratic:

$$X^2 [dt(s+l+t) - 2lt(d-1)] + X [-cd(s+l+t) - cd^2t] + c^2d^2 = 0$$

and solved for X by the quadratic formula, using the smallest positive solution as the maximum throughput.

4. NUMERICAL RESULTS

Ten system configurations were considered: 1, 2, 4, 6, and 8 control modules per PP, with 2 or 4 disks per control module. Each configuration contained one PP. Configurations are designated by the total number of control modules and disks: a system with 4 control modules each having 4 disks is designated as a 4 x 16 system. Sixteen transfer block sizes were considered for each system configuration: from 4 to 64 kilobytes with an increment of 4 kilobytes.

Two results are shown graphically for each system configuration: maximum throughput in requests per second and maximum throughput in megabytes per second. The latter is calculated by multiplying the former by the transfer block size and is a more useful statistic since it gives an indication of the maximum rate of information transfer. These results are graphed in figures 3 to 6.

As expected, the maximum throughput in terms of requests per second decreased with increasing block size and the maximum throughput in terms of bytes per second increased with increasing block size. The maximum throughput of the IPI channel between the PP and the control module never exceeded 6 megabytes/second. Since it is rated at 10 megabytes/second, the maximum utilization of the channel is approximately 60%. This confirms a rule of thumb which states that the system performance drastically deteriorates when the utilization of the channel approaches 60%. The graphs also demonstrate that the channel supports up to four control modules quite efficiently. However there is only marginal benefit in adding additional control modules beyond four, especially with four disks per control module.

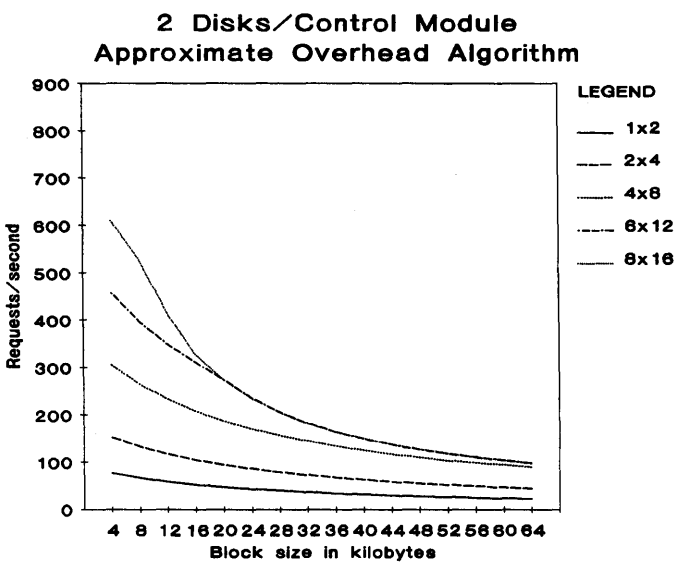


Figure 3

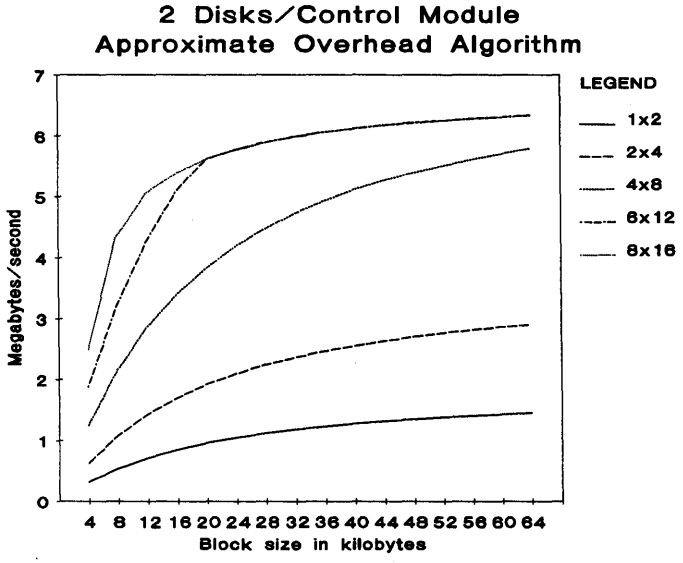


Figure 4

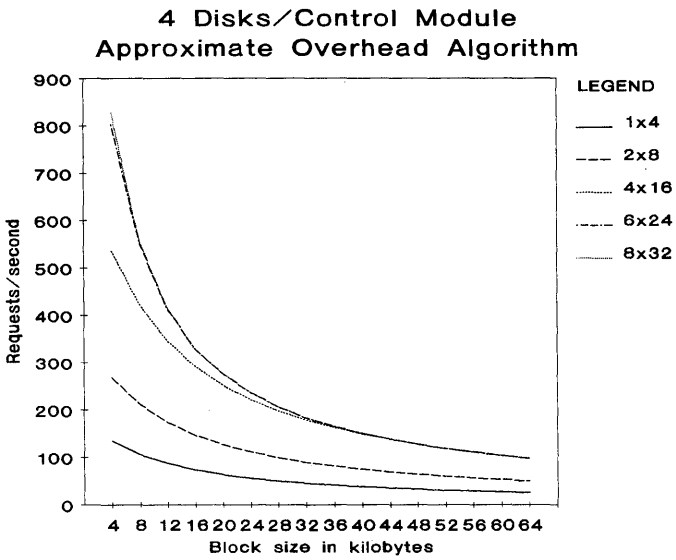


Figure 5

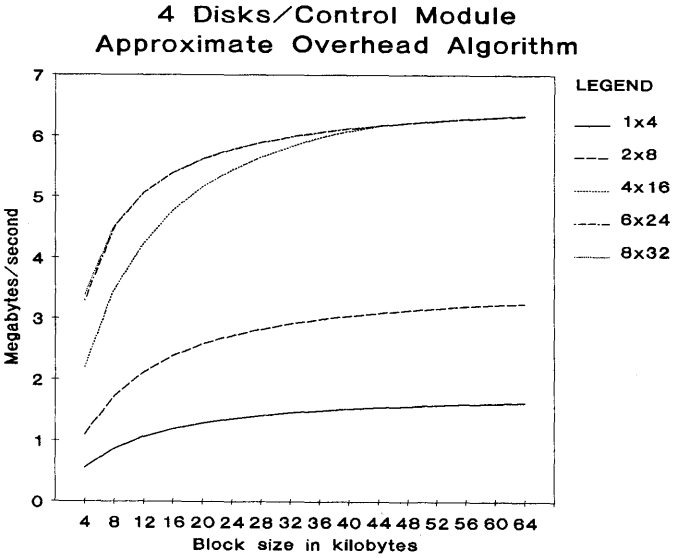


Figure 6

5. GENERALIZATION TO OTHER SYSTEMS

This method of estimating the maximum throughput of a disk subsystem can be generalized to other systems provided that the service demands in the resultant queueing models of those system are either constant or closed-form functions of system throughput. The formula for the disk service demand used in this study was taken from [1]. This book provides closed-form formulae for the following cases: single-path non-RPS architecture with one level of control devices and single-path RPS architecture with one or more levels of control devices. Once these formulae are known, the maximum throughput can be found by following the method outlined in Section 3.

This study also incorporates an analytical expression for the PP based I/O driver overhead. Often this data is not available. We were able to obtain this information directly from the developer of the I/O driver who had done detailed path timing analysis [2]. For other architectures, I/O driver overhead may occur in the CPU or in other elements of the I/O subsystem. As always in performance evaluation, the difficulty lies in understanding where the significant contributions to I/O processing time actually occur.

In general, this method does not work for systems that employ multipathing. For these systems, the proportion of the requests of a disk that use a given path is usually unknown. An iterative algorithm must be used to find the proportioning of the requests of a disk among possible channels and the service demand of the disk cannot be expressed as a closed-form function of the system throughput. However, there are some multipathed systems where the proportioning of the requests of a disk across channels is known a priori. An example is a system where all possible connections between devices on one level and devices on the next level are made. In this situation the requests of a disk are distributed uniformly among all possible channels, allowing the disk service demands to be expressed as a closed-form function of system throughput.

The formula used to calculate the disk service demand assumes a single level of control devices. There are actually two levels of control devices in the system but because the control module is buffered, the disk only "sees" one level of control devices. In general, the formula used to calculate disk service demand should be based on the number of levels of control devices that a disk "sees". Any further levels that are buffered away should be treated as separate service centers.

In this study, the I/O activity was assumed to be balanced across all of the disks. Had this assumption not been made, it would have still been possible to estimate the maximum throughput. The computations are then more difficult as an expression for the service demand at every disk must then be found and the disk throughputs summed.

A thorough treatment of RPS reconnect delays is found in [6]. Another approach to modelling multilevel I/O subsystems is the method of surrogate delays detailed in [7].

6. LIMITATIONS

This study only addresses the saturation case where there are always I/O requests pending for each device. As soon as one request completes, another is ready to be processed. There is also no consideration given to the CPU time required to generate and post an I/O request. For example, if the CPU time for each I/O request were 2 milliseconds which is not unreasonable for this class of system, the maximum I/O rate would be limited to 500 requests/second because of CPU saturation.

Another assumption made is that the distribution of I/O requests is uniform across all devices and control modules. Although this is almost never achieved in practice, it is a reasonable assumption because we are trying to determine the maximum possible throughput for the disk subsystem which occurs when the load is evenly distributed. As was previously noted, the solution technique used here is still usable without a uniform distribution albeit with additional computational complexity. Other factors which affect these results are the mean seek time and the distribution of seek time. In this study, we have assumed an average seek time of 11 milliseconds based on measured results from benchmarks. However this is highly dependent on the workload type and on the amount of tuning done on file placement.

Further studies are planned to determine the effects of these factors.

7. CONCLUSIONS

A method based upon queueing network theory of estimating the maximum throughput of the CYBER 930 Disk Subsystem has been devised. A queueing model of the system was constructed and parameterized. The computations involved with this model are easy and can be done by hand if necessary.

The results were graphed for different disk configurations and different transfer block sizes. They predict that no significant benefit is to be gained by having more than four control modules per peripheral processor. In fact the graphs for the exact overhead algorithm show that performance actually degrades if there are more than four control modules on a channel.

As well, they predict that the maximum throughput in terms of megabytes per second increases with increasing block size and that the maximum utilization of the control module - peripheral processor channel is approximately 60%, both of which are in agreement with empirical data. However this method has not been otherwise validated.

From this analysis, the expected peak disk transfer rate on any individual channel is about 5.5 megabytes per second for FSD-2 class disks. A typical large CYBER 930 configuration has four disk channels. This means that an aggregate disk transfer rate of 22 megabytes per second could be attained. In fact as was discussed in Section 6, we would expect the CPU to become saturated well before this throughput level could be reached.

This method can easily be adapted to work for many other disk systems.

8. REFERENCES

[1] Lazowska, E.D., J. Zahorjan, G.S. Graham, and K.C. Sevcik: Quantitative System Performance - Computer System Analysis using Queueing Network Models, Prentice-Hall, Englewood Cliffs NJ, 1984.

[2] Rowe, C.D.
Private Communication.

[3] Hunter, D.: Modelling Real DASD Configurations, Applied Probability - Computer Science: The Interface, Vol. I, Birkhauser, 1982, 451-468.

[4] Wilhelm, N.C.: A General Model for the Performance of Disk Systems, JACM 24,1 (January 1977), 14-31.

[5] Atwood, J.W. and K.-C. Yu: An Empirical Study of Disk Subsystem Parameters, University of Texas at Austin Computation Center Note CCSN 43, July 1980

[6] Buzen, J.P. and A.W. Shum: A Unified Operational Treatment of RPS Reconnect Delays, Proc. 1987 ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems, May 1987

[7] Garner, R.L., P.A. Jacobson, and E.D. Lazowska: The Method of Surrogate Delays: Simultaneous Resource Posession in Analytic Models of Computer Systems, University of Washington Department of Computer Science Technical Report No. 80-04-03, 1980.

APPENDIX A - A MORE ACCURATE EXPRESSION FOR PP OVERHEAD

The algorithm used in this paper to calculate the peripheral processor overhead is fairly accurate for small block sizes but underestimates the overhead for larger block sizes. The reason for this is that the peripheral processor deselects from a control module after three sectors of 2048 bytes each have been transferred in order to multiplex to other units. If more than three sectors need to be transferred, the control module needs to be reselected, causing an increase in overhead. A more accurate algorithm to calculate the PP overhead is given by [2] and shall now be presented.

In this algorithm the PP overhead can be divided into four components: I/O setup, 6 kilobyte transfers, remainder transfers, and response to the CPU.

Let B be the block size in kilobytes,
P be the page size in kilobytes,
c be the number of controllers,
d be the number of disks per controller.

The initialization time is the time taken for initialization of the request.

$$\textit{Initialization time} = \frac{55+10\times cd}{2} + 230 + 13\times\left\lceil\frac{B}{P}-1\right\rceil \mu s$$

$$= 257.5 + 5cd + 13\times\left\lceil\frac{B}{P}-1\right\rceil \mu s$$

The 6 kilobyte setup time is the PP overhead associated with a transfer of a series of data bursts of 6 kilobytes each.

$$\textit{6KB setup time} = \left\lfloor\frac{B}{6}\right\rfloor\left(\frac{55}{2} + 10c + 75 + (3\times 85) + 25\right) \mu s$$

$$= \left\lfloor\frac{B}{6}\right\rfloor (382.5 + 10c)\ \mu s$$

The remainder setup time is the PP overhead associated with the transfer of the remainder of the block after no more 6 kilobyte burst transfers are possible.

$$\textit{Remainder setup time} = \frac{55}{2} + 10c + 75 + 85\times\left\lceil\frac{B}{2}\ mod\ 3\right\rceil + 25\ \mu s$$

$$= 127.5 + 10c + 85\times\left\lceil\frac{B}{2}\ mod\ 3\right\rceil \mu s$$

The response to the CPU is the time taken by the PP to tell the CPU that a transfer has been completed.

$$\textit{CPU response} = 177.5 + 10c\ \mu s$$

The total PP overhead is then found by summing the four components.

$$\textit{PP overhead} = 562.5 + 20c + 5cd + 13\times\left\lceil\frac{B}{P}-1\right\rceil + 85\times\left\lceil\frac{B}{2} mod\ 3\right\rceil +$$
$$\left\lfloor\frac{B}{6}\right\rfloor\times(382.5 + 10c)\ \mu s$$

Note that this expression does not include the data transfer time on the control module to central memory IPI channel path. The total PP service demand is the sum of the PP overhead and this data transfer time.

A more accurate estimate of the maximum throughput of the peripheral processor can be found by using this estimate of the PP overhead in the formula for the maximum PP throughput in section 3.1. This has been done and the results from these new calculations are shown in figures 7 through 10.

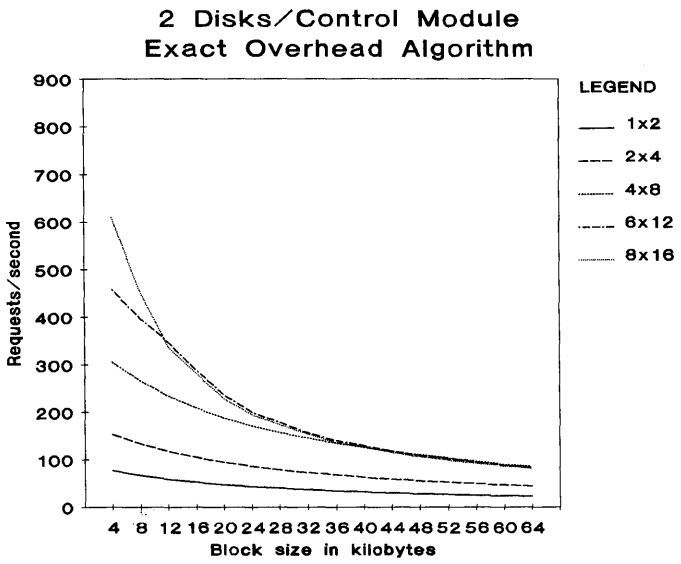


Figure 7

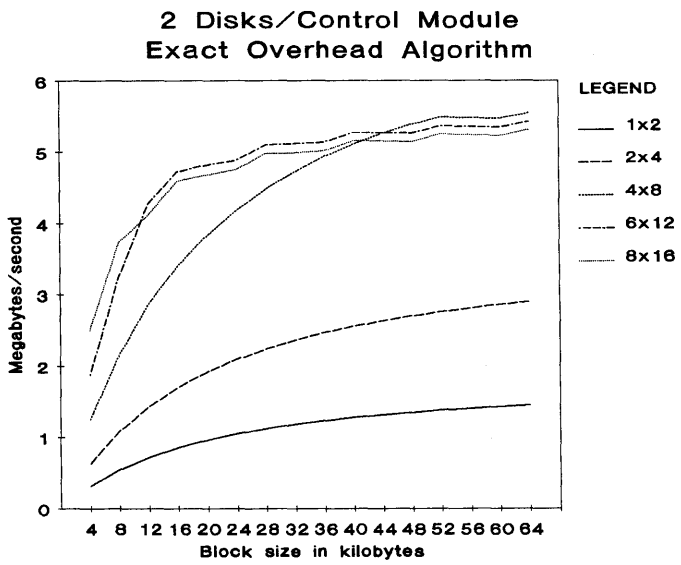


Figure 8

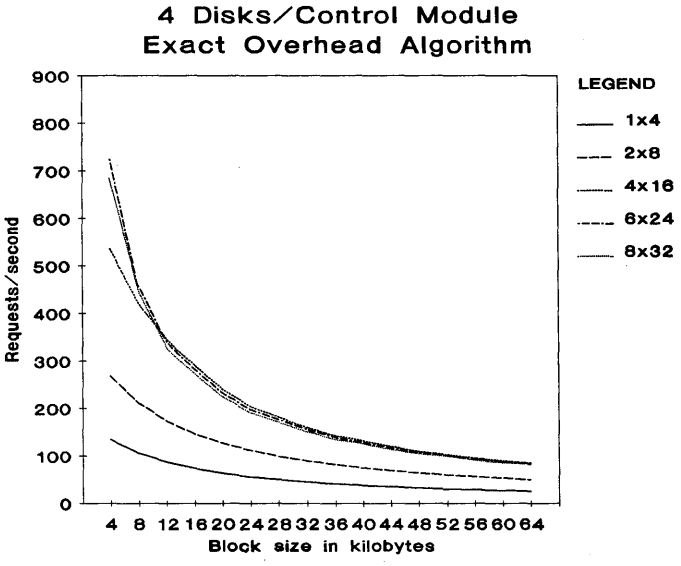


Figure 9

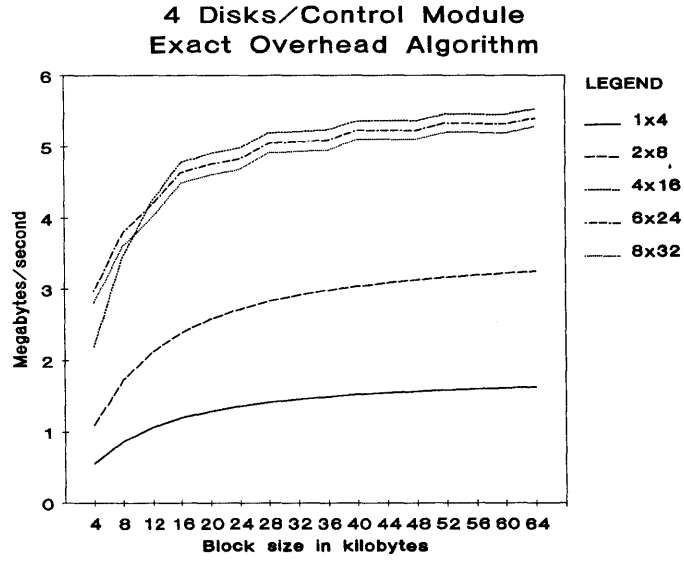


Figure 10

PERFORMANCE ARCHITECTURE COMPARISON OF VECTOR PROCESSORS ON LARGE SCIENTIFIC AND COMMERCIAL MACHINES

Dr. David S. Lindsay
National Advanced Systems

ABSTRACT

Using a benchmark described in a companion paper, this paper compares a Cray-XMP (an "engineering/scientific" machine) with an IBM 3090-VF and an NAS AS/XL, the latter two being "commercial" machines with vector processors.

The benchmark technique reveals such details as (1) the vector vs. scalar speed of each machine for various operations and vector lengths (and, therefore, what minimum vector length is required for the vector processor to surpass scalar performance); (2) the ability of vector processors to "chain"; and (3) the pipelining capability of scalar machines, as revealed by performance tests of "loop unrolling".

1. INTRODUCTION

A companion paper (Ref. 1) has described the design of a benchmark for vector architecture and performance comparison. That paper noted that performance is an even more important consideration for a vector processor than for a "normal" processor, since the sole motivation for vector processing is improved performance. With the advent of vector processors on general-purpose machines such as the IBM 3090, it is becoming imperative that CPE professionals adequately understand the performance of vector processors.

Briefly reviewing the benchmark approach described in Reference 1, the following characteristics are of central importance to vector performance:

1. Vector speed vs. vector length;

2. Vector vs. scalar performance;

3. Sensitivity to data caching;

4. Determining the processor's ability to "chain" (vector chaining is analogous to pipelining in a scalar machine--the ability to execute more than one vector operation simultaneously. See Ref. 1).

This list is not exhaustive, but it represents an essential first step in understanding vector performance. Naturally, these metrics may depend on what particular vector operation is being considered. Thus, it is important to check several operations, including the ones that will actually be used in practice. The rest of this paper presents such data for several important vector operations on a Cray-XMP, an IBM 3090-VF, and an NAS AS/XL.

In order to compare vector and scalar performance, this paper also reports results from the same benchmark in scalar mode. As pointed out in Reference 1, "loop unrolling" (described more fully in the next section) can be an important technique for improving scalar performance. Therefore, results from tests with unrolled loops are also presented in this paper.

This paper is organized as follows: Sections 2 through 9 present measurement data from several different vectorizable operations, one operation per section; Section 10 presents some general conclusions about the three machines measured; Sections 11 and 12 present recommendations for users and manufacturers of vector processors, based on the measurement data; and Section 13 presents concluding remarks.

The measurement data show that many of the performance features of vector processing are common to all the tests. Therefore, this paper discusses these common features at length in Section 2 (which also describes the first test); successive sections describe only differences from the properties of Section 2, and are thus considerably shorter.

2. VECTOR ASSIGNMENT

This test measures the CPU speed of the following FORTRAN construct (or its vectorized equivalent):

```
      Do 2 i = 1, N
2     X(i) = Y(i)
```

Surprisingly, although this is a very simple test, its results yield most of the relevant information about vector and scalar performance. Measurement data are now presented:

2.1 CRAY-XMP RESULTS

Figure 1 presents the results of this test on a new Cray-XMP/416. (The significance of "new" is that the machine had a cycle time of 8.5 nanoseconds; most previous Cray-XMP's had 9.5 nanosecond cycle times). The "416" part of the designation specifies four CPU's and 16 mega-words of memory. Since this benchmark requires less than one mega-word of memory.and runs in a single CPU, the results would presumably be the same for other configurations of Cray-XMP CPU, so long as the cycle time remained the same (8.5 nanoseconds).

All figures in this paper have the same axes and scales as Figure 1, except where specifically noted in the text; the horizontal axis represents the logarithm (base 10) of the vector length, so that the origin (0) represents a vector length of 1, "2" represents a vector length of 100, etc. The software that created the graphs extended the horizontal axis to 6 (i.e., vector length 1,000,000), although vector lengths actually ranged up to a maximum of 200,000 in these tests.

For uniformity, the vertical axes extend to 110 million operations per second (MOPS), where for the purposes of the graphs in this paper, an "operation" is one unit of whatever the test measures, no matter how complicated. In Figure 1, an "operation" is a complete assignment--that is, the load and store are not counted separately.

Turning now to the data displayed in Figure 1, the scalar performance figures only extend to a vector length of 500 (rather than 200,000 as do the vector results) because of the expense of running on a Cray coupled with the expectation of seeing no new features in the data beyond length 500 for scalar loops.

Vector code was generated by a vectorizing compiler (CFT Version 1.14E), and shows many interesting features. Although the test was simple (merely copying a vector from storage to storage), the qualitative features of this test are similar to the other tests on the Cray-XMP, as will be seen below. The main features of interest from Figure 1 are:

1. The Cray-XMP's maximum performance on long vectors was roughly 85 MOPS, although the 8.5 nanosecond cycle time of that machine should have permitted 118 MOPS. Some of this discrepancy might have been due to running in a busy environment (for example, the performance dip between vector lengths 2,000 and 20,000 was probably due to timing noise or higher than average interference from other jobs).

But an additional source of overhead was the limit on the vector registers of the Cray-XMP: only 64 elements. Therefore the Cray hardware cannot process a vector longer than 64; longer vectors are split into 64-element pieces by the software, and processed in a loop (referred to as a "sectioning loop"). There is evidently enough loop overhead, either in the hardware or the software, that the theoretical maximum speed of 118 MOPS was not achieved in this test (or any other test in this set of benchmarks),

2. Although performance did not come close to the theoretical maximum of 118 MOPS, it certainly exceeded half that value, indicating that the machine was able to process more than one operation (including both a load and a store) in two machine cycles. This performance figure demonstrates that the Cray-XMP was able to "chain" its vector instructions (see Ref. 1, Fig. 3)--the processor could overlap a vector load and a vector store.

3. Figure 1 also shows that very long vectors (length 1,000 or more) were required to reach the processor's maximum speed; a vector length between 50 and 100 was required to achieve half of that performance.

4. Interestingly, vector performance exceeded (or at least equaled) scalar performance for <u>all</u> vector lengths; and scalar performance rose to about 4 MOPS for a vector length of 10 or greater, and was close to that value even for a vector length of five. Therefore scalar performance seems to approach a maximum for small loop counts (5 or 10), whereas vector performance requires vastly longer arrays before its full promise can be reached (in fact, a strict interpretation of "full performance", namely one operation per machine cycle, was <u>never</u> reached for this or any other test).

2.2 SCALAR LOOP UNROLLING

The scalar data reported in Figure 1 appear to be too pessimistic--scalar performance of two or even three times that shown has been reported from other Cray performance tests (Ref. 2). This discrepancy suggested the use of "loop unrolling". As applied to the assignment test, loop unrolling would restructure the FORTRAN source to be:

```
      Do 1 i = 1, N, 2
      X(i) = Y(i)
1     X(i+l) = Y(i+l)
```

Such a construct places two assignment statements (referencing adjacent vector elements) in each pass through the loop, and therefore executes the loop half as many times. (Some additional logic is required to process the general case where N, the vector length, may be odd.) Performance improvement in this unrolled loop arises from two related effects:

1. There is half as much loop overhead;

2. There is twice as much "useful work" per branch instruction, potentially making pipelined machines operate much more efficiently.

It may seem that loop unrolling is clumsy, seldom used in real programs, and therefore only of academic interest; but, in fact, there are several reasons for believing that unrolled loop performance is more valid than that of the simple, straightforward scalar loop used in this assignment test:

1. The MFLOPS rate ("millions of floating point operations per second", not necessarily the same as MOPS in this paper) usually seen on the Cray-XMP in scalar processing is much greater than four (Ref. 2), suggesting that the simple assignment test is not an accurate gauge of real performance on real work;

2. "Real" programs have much more complex loops than the one used here--they have many instructions per loop, thereby drastically reducing the percentage effect of loop overhead, while simultaneously increasing the efficiency of heavily pipelined machines (and those high-performance machines are exactly the ones most likely to feature vector processors);

3. Some advanced compilers, such as the latest Cray compiler (not used in these tests), are reputedly able to generate unrolled code for the short scalar loops that could benefit from such a technique.

For all those reasons, the benchmark now includes (in addition to the standard "rolled" loops) unrolled scalar loops for all tests. There are three sets of unrolled tests, with 2, 5, and 10 iterations per loop. Figure 2 presents results from these unrolled tests, in addition to those shown in Figure 1. Again, scalar results from the Cray were only run for vectors up to length 500, due to the difficulty of arranging for Cray time coupled with the expectation of seeing no performance change for scalars beyond length 500.

Figure 2 also presents data from an IBM 3090-VF and an NAS XL (discussed below). Because of limitations in the software used to prepare the graphs, the legend for the curves is rather brief. The legends mean:

- Sca--Scalar;

- 2--Unrolled scalar loop with 2 iterations per loop;

- 5--Unrolled scalar loop with 5 iterations per loop;

- 10--Unrolled scalar loop with 10 iterations per loop;

- Vct--Vector code (generated by compiler or the Cray, and by hand for the IBM 3090 and the NAS XL);

- VAST--Vector code generated by the VAST pre-processor (NAS XL results only).

Figure 2 shows that a dramatic speed improvement was obtained on the Cray by loop unrolling: performance for longer loops climbed from 4 to 8, 15, and 19 million assignments per second when the loops were unrolled by 2, 5, and 10 iterations. Perhaps performance would improve further with more loop unrolling.

Note an important feature of the Cray unrolled scalar loops: they required long "vectors" for high performance. For example, with 10 iterations per loop, 50% of maximum performance was not reached until the "vector length" reached 20. In this sense, unrolled scalar loops on the Cray behave like vector operations.

Loop unrolling presents performance in a new light: while it makes scalar performance look better, it thereby worsens the machine's relative vector performance. If the Cray's scalar speed is believed to be about 4 MOPS (as it appears to be in this test without loop unrolling), the vector processor improved performance 21 times (to 85 MOPS); but if improved scalar coding techniques (such as loop unrolling) can boost scalar speed to 19 MOPS, then vector speed-up becomes only a factor of 4.5, considerably less stunning (but nonetheless impressive).

But regardless of the vector speed-up factor, the fact remains that the Cray's vector processor was always faster than its scalar processor on assignments of any vector length. So there was never a penalty associated with using the vector processor for assignments.

2.3 IBM 3090-VF RESULTS

Figure 2 also presents the results of the assignment test on an IBM 3090-200/VF. This machine was a dual processor, but since the benchmark only runs in one CPU, it is really a test of only a single CPU and its vector processor. The dual-processor nature of this machine should not affect the results in this paper.

Scalar code was generated by the IBM VS-FORTRAN 1.4.1 compiler; vector code was hand-assembled (the version of VS-FORTRAN that we had at the time the tests were run generated incorrect vector code in some cases).

There are several interesting features of the IBM 3090 results shown in Figure 2:

1. The maximum vector processing rate was about 20 MOPS, about the same as the Cray's unrolled scalar loop performance. But the IBM 3090's cycle time was 18.5 nanoseconds, so one operation per cycle would have yielded a processing rate of 54 MOPS. Thus, the IBM 3090 was not able to process the vector assignment in one cycle per element; it only reached about one REAL*8 assignment in 2.5 cycles.

The vectorized machine code consisted of a 128-element vector load followed by a similar vector store in a sectioning loop (the IBM 3090 that we measured had 128-element vector registers). Since the resulting performance was somewhat less than one assignment in two cycles, the IBM 3090 was not capable of vector chaining. This conclusion is also supported by the rest of the benchmark tests. The lack of vector chaining on a machine with vector registers is a particularly serious shortcoming, since extra machine cycles are therefore required to load and store data.

2. Unrolling the scalar loop made little difference for the IBM 3090; perhaps that machine is not sufficiently pipelined to take advantage of several instructions in a loop.

3. Unlike the Cray's scalar processor, the IBM 3090's scalar processor appeared to reach 50% of its maximum performance at a "vector length" of about two. There is evidently far less loop overhead (either hardware or software) on an IBM 3090 than on a Cray.

4. A significant difference between the results from the IBM 3090 and the Cray was the reduction in IBM 3090 performance for vector lengths above 2,000. This effect is probably due to exhausting the IBM 3090's cache--when the operands do not fit in cache, the vector processor must access main memory, resulting in lower performance. (The Cray does not have cache, so no such effect is visible in the Cray data.)

5. As was the case for the Cray, the IBM 3090 required long vectors to reach its maximum performance: performance did not achieve 50% of its maximum value until the vector length was about 30, and was still rising at a vector length of 200.

6. The IBM 3090's vector processor was slower than its scalar mode when the vector length was less than 20 or 30 (the exact length depends on the level of scalar loop unrolling that is considered). This vector performance is considerably less impressive than the Cray's, which appeared to be faster (or at least no slower) in vector mode for any vector length.

7. A related feature of Figure 2 was that the maximum vector speed-up was much smaller for the IBM 3090 than for the Cray: a factor of slightly less than two.

8. The IBM 3090's scalar performance was higher than the Cray's unless loop unrolling of five or ten iterations was used.

There are two possible architectural reasons for the Cray's relativeiy poor scaiar performance (compared with the IBM 3090's) on rolled loops:

1. The Cray has 15 cycle memory and no cache. Thus loads from memory are forced to consume 15 machine cycles in all cases; the only hope for increased speed is the possibility of executing some other useful work while a load is in progress.

2. The Cray does not have "multiple instruction stream" capability, the ability to begin execution of both possible instruction streams before the outcome of a branch is known. The IBM 3090 and the NAS XL have such a capability, reducing the impact of a branch instruction on their pipeline performance.

Both of those architectural differences could make the Cray's scalar processor highly dependent on loop length.

2.4 NAS XL RESULTS

Figure 2 also presents benchmark results from an NAS AS/XL VIOO, a quadruple processor. As with the Cray and IBM 3090 tests, the benchmarks ran on one CPU; presumably the others would show identical results. This NAS XL CPU included a vector processor and had an 18 nanosecond clock, very close to the IBM 3090's 18.5 nanosecond clock. Thus any significant differences between the IBM 3090 and NAS XL performance reflect a difference in the number of machine cycles required to perform an operation--an architectural difference.

Unlike the Cray and the IBM 3090 vector processors, the NAS XL is a "memory-to-memory" machine: it has no vector registers. Thus one would expect that the NAS XL would not exhibit the "sectioning" overhead that was evident in both the Cray and IBM 3090 data.

The NAS XL scalar results in this paper were generated by runs from the same object code as the IBM 3090 runs (the NAS and IBM machines used in this study were in the same data center, and are instruction-compatible for all scalar instructions).

The figures also present results from both hand-assembled vector code and FORTRAN code generated by the pre-processor VAST (a product available for the NAS XL that processes FORTRAN code and inserts library calls to implement vectorization). The machine-generated code obviously had more overhead than the hand-assembled code, probably due to subroutine linkage. Both sets of code produced the same maximum performance. The graphs suggest the following observations:

1. The maximum speed of the NAS XL vector processor (for either method of code generation) was about 53 MOPS, which was very close to its theoretical limit (one operation per 18 nanosecond machine cycle, or 56 MOPS). This speed demonstrates that the NAS XL was capable of chaining the load and the store to achieve nearly one assignment per machine cycle.

2. The effects of cache are visible, just as they were for the IBM 3090; but the NAS XL's vector performance decreased more than the IBM 3090's for long vectors, probably due to the NAS XL's maximum performance being higher. Interestingly, the performance without cache (i.e., for the very longest vectors) of both machines was about the same.

Strangely, the vector length at which cache degradation began was not the same for the hand-coded and the machine-generated code. Since they both ultimately used the same machine op-code, this difference is at first sight puzzling; it seems to be due to details of the cache replacement algorithm, which depend on the real address of the data. On different runs, the MVS operating system (which employs virtual addressing) presumably mapped the arrays to different real addresses, causing some variation in the maximum working set size permitted by the cache replacement algorithm.

3. On the standard (rolled) scalar loops, the NAS XL achieved about the same performance as the IBM 3090, as one might expect from their nearly identical cycle times. But there was a surprise:

4. The NAS XL exhibited much greater performance improvement on loop unrolling than did the IBM 3090. Rolled scalar loop performance reached 9 MOPS on the NAS XL, whereas its unrolled scalar loop performance reached 23 MOPS, higher than the Cray's unrolled performance, more than a factor of two above the IBM 3090's unrolled performance, and in fact higher than the 3090's vector performance! Since both the NAS XL and the IBM 3090 have nearly the same cycle times, one must conclude that the NAS XL's scalar processor is capable of more instruction overlapping--and this capability apparently becomes evident when branches do not interfere with the machine's pipelining.

5. Like the IBM 3090, the NAS XL achieved 50% of its maximum scalar performance with very short loops: five or less. This good scalar performance on small loops is in sharp contrast to the Cray's scalar overhead, as mentioned above.

6. As was the case for both the Cray and the IBM 3090, the NAS XL required long vectors to achieve its maximum performance--vector lengths in the range 500 to 2,000. Half of the maximum was obtained between 30 and 100 (depending on the method of code generation). Vector break-even occurred for vectors of length 5 to 20 for rolled scalar code, and 20 to 70 for unrolled scalar code.

Overall, the most startling result of the vector assignment test may have been the scalar results: two machines with nearly identical cycle times (the IBM 3090 and the NAS XL) exhibited such drastically different performance; and the NAS XL's scalar processor was faster than the Cray's, which had half the NAS XL's cycle time. It is evident that the architectures of these three machines are very different.

In conclusion, even though the assignment test might not appear to measure anything "useful" (one hardly buys a vector processor to move data from one storage location to another), it nevertheless conveys much useful information, such as:

* The ability (or lack thereof) of the vector processor to "chain" operations;

* The extent of the scalar processor's pipelining capability;

* The ability of the vector processor to approach the theoretical maximum performance derived from its cycle time;

* The minimum vector length required for the vector processor to surpass the scalar processor's performance (i.e., the "vector break-even length");

* The problems that some machines encounter with cache degradation for long vectors.

These factors will now be described for arithmetic operations.

3. VECTOR MULTIPLY

The previous section's discussion of the "Vector Assignment" test was very long, since it included many descriptions of vector and scalar processing issues, along with a discussion of the way the graphs in this paper present data. With that background now established, the sections describing the other vector tests can be much shorter. This section describes the "Vector Multiply" test.

By contrast with the vector assignment test, the vector multiply test actually performs some "useful" work: REAL*8 multiplies. The test times the following FORTRAN loop (or its unrolled or vectorized equivalent):

```
      DO 3 i = 1, N
3     X(i) = Y(i)*Z(i)
```

Figure 3 presents timing data from a Cray, an IBM 3090, and an NAS XL. The main points to observe are:

1. Figures 3 and 2 are quite similar. In particular, both the Cray and the NAS XL achieved about the same vector performance on multiply as on assignment, indicating that both machines are capable of vector chaining. That is, adding an additional memory reference and a multiply made little or no difference in the execution time! That startling result illustrates the power of vector chaining.

2. The IBM 3090's vector processor, however, slowed to somewhat less than one operation in three machine cycles, indicating that it does not chain--the load, multiply, and store operations are not overlapped and proceed sequentially.

3. Loop unrolling significantly improved both the Cray's and the NAS XL's scalar performance, but had little effect on the IBM 3090's; this result was similar to the vector assignment test (Figure 2). A closer comparison of Figures 2 and 3 shows that for all three machines, unrolling the multiply loops had less effect than unrolling the assignment loops. This result is easy to understand--the multiply test has more instructions in its scalar loop, decreasing the loop overhead (and increasing the pipelining) for the scalar loop. Therefore, the multiply loop is more efficient than the assignment loop, so unrolling yields less (but still significant) performance improvement.

Apart from those differences, the results shown in Figure 3 are remarkably similar to those of Figure 2; again, the NAS XL scalar performance on unrolled loops was about the same as the maximum IBM 3090 vector performance, and preserved its peak value to much shorter vectors. Also, the Cray's scalar processor required by far the longest "vectors" to reach 50% of its maximum performance.

4. VECTOR ADD

The vector add test times the following construct:

```
            Do 4 i = 1, N
4           X(i) = Y(i) + Z(i)
```

The results from the vector add tests are reproduced in Figure 4. They were essentially identical to those from the vector multiply. This important result should be stressed --vector adds and vector multiplies took the same time on the three machines tested. This performance equality exists whether or not the adder and multiplier are equally fast in scalar mode; the essential point of vector operation that should be clearly understood is that properly implemented pipelining produces one result per cycle, regardless of the length of the operation involved, providing that the vector length is large enough to overcome start-up effects. This form of pipelining within an arithmetic operation is sometimes called "segmentation" (particularly in CDC and Cray parlance).

Note that the scalar tests shown in Figure 4 were slightly faster on the IBM 3090 and the NAS XL than the scalar results of the multiply test (Figure 3); the adders on these machines are slightly faster than the multipliers, so that scalar adds go a bit faster.

5. VECTOR DIVIDE

At this point, the reader might be under the impression that all vector instructions proceed at the same rate, especially on machines with vector chaining. Unfortunately, such is not the case: divide operations present a serious problem to machine designers, since the divide operation does not appear to be segmentable (it is usually implemented as an iteration, thereby using the same hardware gates for several cycles). The ability to segment an arithmetic operation is crucial to vector performance, since without segmentation the processor is unable to deliver one operation per cycle and the advantage of vectorization is essentially lost.

Figure 5a presents timing data from the vector divide test, which times the following construct:

```
        Do 5 i = 1, N
5       X(i) = Y(i)/Z(i)
```

The figure shows that vector divide performance is drastically below that of the previous vector operations. Divide performance is so low that Figure 5b has been prepared with an expanded vertical scale (extending to 30 MOPS instead of 110 MOPS).

Figure 5b shows that loop unrolling only helped the Cray's scalar performance (perhaps because of the Cray's architectural differences mentioned above). On the IBM 3090 and the NAS XL, divide was such a slow operation that loop overhead was insignificant (so unrolling the loop did not materially reduce the overhead); and since the machines' divide instructions evidently could not be segmented, there was no pipeline improvement available to loop unrolling.

The machines' maximum vector speeds for divide were:

* Cray-XMP -- 4.7 cycles;
* IBM 3090-VF -- 20 cycles;
* NAS AS/XL -- 6 cycles.

Programmers frequently avoid divide instructions in order to improve performance. For example, "Gaussian Elimination", a technique often employed in matrix inversion, requires among other things that each row of the matrix be divided by the value of its first element. This "normalization" is usually accomplished by first inverting the first element, then multiplying by that inverse using either a scalar loop or a vector multiply operation. In that way, a vector divide (or a scalar loop full of divides) is avoided.

Machine designers frequently use hardware monitors to capture op-code frequencies; they therefore see few divide instructions, and conclude that divide is a rare instruction, so building a fast divider is unnecessary. The result may be a self-fulfilling prophecy: divide instructions remain slow and seldom used.

6. ADD UP A VECTOR

A frequent vector operation is summation--not the equivalent of a vector add, but rather summing the elements of a vector:

```
        SUM = 0
        Do 6 i = 1, N
6       SUM = SUM + X(i)
```

Summation is a component of the "Dot Product" operation, frequently used in engineering. As Reference 1 points out, summation or "Add up a Vector" is not truly vectorizable (in the sense of being able to pipeline one add per machine cycle) unless the complex technique of "partial sums" is used. With partial sums, vector performance improves at the expense of evaluating the summation in a different order, thereby possibly yielding a slightly different result. Figure 6 presents the measurement data from this test.

Figure 6 is different from the earlier tests in several ways:

1. The Cray results show the highest vector performance of any of these tests--110 MOPS, achieved using a partial sum number of eight (i.e., eight separate add chains; see Reference 1). This peak performance was obtained at a cost, however: 50% of the maximum was not obtained until the vector length reached 500, and performance was still rising at a vector length of 20,000! For more modest vector lengths, performance was considerably lower; at a vector length of 50, for example, only about 15 MOPS was obtained. But at that vector length, both the IBM 3090's and the NAS XL's scalar speed exceeded the Cray's vector speed!

Paradoxically, although the Cray achieved its highest performance on this test, the scalar speed exceeded the vector speed for vectors shorter than about 20.

2. The IBM 3090 results also showed that machine's highest vector performance--about 40 MOPS, almost twice the earlier maxima. This result was achieved by the hardware's partial sum mechanism, employing a partial sum number of four (i.e., four separate add chains). But unlike the Cray results, the IBM 3090 performance reached 50% of its maximum at the much more modest and achievable vector length of 70 (which was only slightly above the vector break-even length for unrolled loops). Note that at that vector length, the IBM 3090 performance exceeded the Cray's.

3. The NAS XL performance was also noteworthy in that the unrolled scalar loop performance was substantially better than the vector performance! The reason is twofold: (1) the NAS XL vector processor does not use partial sums, and (2) its adder is capable of operating once every 1.5 machine cycles. With a 1.5 cycle adder, a sufficiently unrolled scalar loop would be capable of 37 MOPS (which was achieved, as Figure 6 shows); but the vector processor, which can only issue instructions on cycle boundaries, was therefore limited to two cycles per add, yielding 28 MOPS (also achieved, as shown in the figure).

Because the NAS XL does not use partial sums, it does evaluate the summation in the order specified in the FORTRAN code. For some critical numerical computations, the order of summation may be important. For that reason, the Cray compiler has a directive that forces computations to proceed in the order implied by the FORTRAN source, and therefore eliminates vectorization of such operations as vector summation and dot product. The programs that require such treatment would also preclude vectorization on the IBM 3090, but could proceed without problems on the NAS XL.

Note also that for vectors shorter than about 200, the Cray was the slowest of the three machines measured.

7. DOT PRODUCT

The dot product construct is frequently encountered in engineering and scientific applications. It can be defined by the FORTRAN loop:

```
        DOTPROD = 0
        Do 7 i  = 1, N
7       DOTPROD = DOTPROD + X(i)*Y(i)
```

Note that "Add up a Vector" is an integral part of "Dot Product". Figure 7 presents measurement data on "Dot Product".

As was the case for the "Add up a Vector" test, the Cray joined the IBM 3090 and the NAS XL in that short vectors were faster in scalar mode. Also, the Cray required a very long vector (about 300) in order to achieve half its maximum vector performance. The performance difficulties of the "Add up a Vector" component of the calculation are evidently also inherent in dot product.

By contrast, the IBM 3090 approached its maximum vector performance at a vector length of about 200; 50% of maximum was achieved somewhere between vector length 20 and 50. Scalar loop unrolling did not assist the IBM 3090, and as above the Cray's scalar processor required much longer vectors than the other machines in order to reach 50% of its maximum rate.

Inquiries to Livermore disclosed that the Cray compiler generates a subroutine call to process dot product, thus encountering significant overhead. For some reason the NAS XL, using the VAST pre-processor (which also generates a subroutine call), did not encounter as much overhead, although the IBM architecture (which is implemented by the NAS XL for scalar instructions) is frequently criticized for its excessive overhead on subroutine linkage--and the Cray operating system is felt to be efficient at subroutine linkage.

When the vector length became long enough for the Cray to approach its maximum speed, it executed at about 95 MOPS, faster than any test other than "Add up a Vector". Yet dot product is more complex: for the Cray architecture, it requires two vector loads, a vector multiply, and a vector summation operation. Evidently all four operations were chained, so that they could proceed simultaneously in a single machine cycle. The Cray's dot product performance is an important illustration of a peculiar property: vector performance is less related to the complexity of the operation than to the characteristics of the machine's architecture.

The IBM 3090 achieved a maximum rate of somewhat more than one operation in three cycles--impressive performance for a non-chaining machine, considering that a load, a multiply, and a summation are required on that machine. Actually, there is an instruction "Vector Multiply and Accumulate" that internally chains the multiplier and the adder of the IBM 3090 (that is, the single op-code exercises both the multiplier and the adder in parallel) so that dot product, which can use that instruction, is optimized. Evidently the machine designers targeted dot product for special treatment.

The NAS XL designers also targeted dot product for special treatment by including a "Dot Product" op code. Nevertheless, full chaining was not achieved (since partial sums were not used); the instruction executed in one operation per two machine cycles.

Those who are interested in the metric "millions of floating point operations per second" (MFLOPS) should note that, because dot product includes both a multiply and an add, the MFLOPS rate is twice the MOPS rate reported here. The maximum MFLOPS rates for the three machines were thus about 185 (Cray), 40 (IBM 3090), and 55 (NAS XL).

8. SCALAR*VECTOR + VECTOR

As mentioned in Reference 1, this set of operations is frequently encountered in matrix inversion codes. The following FORTRAN loop describes these operations:

```
      Do 8 i = 1, N
8     X(i) = X(i) + A*Y(i)
```

Where X and Y are vectors as above, but A is a scalar.

This test should be especially interesting to CPE analysts since it comprises the bulk of the LINPACK performance test (Ref. 3); many "engineering/scientific" customers are heavily influenced in their acquisitions by LINPACK performance figures.

The results of the "Scalar*Vector + Vector" test yield exactly the same information as the LINPACK benchmark--more in fact, since LINPACK is usually run only for a few vector lengths (lOOxlOO and 300x300 are the typical matrix sizes). Figure 8 presents the results of these tests. Like dot product, "Scalar*Vector + Vector" contains both a multiply and an add, so its MFLOPS rate is twice the MOPS rate reported in the figures.

The NAS XL was almost as fast as the Cray on this test. The NAS XL's designers also targeted this operation for optimization: they implemented "Scalar*Vector + Vector" as a single machine op-code, with chaining. This peculiarity of the NAS XL hardware demonstrates the fallacy of using this test (or the LINPACK benchmark, which is the same thing) as the sole measure of vector performance: the relative performance of the three machines is thereby presented in a distorted light. Compare the graphs of Figure 8 with the other figures; Figure 8 is atypical in two important respects:

1. The Cray's maximum rate was about 70 MOPS, the lowest of all the Cray vector performance figures;

2. The NAS XL had a special instruction to perform this test, making its performance about 53 MOPS, about equal to the highest of any NAS XL vector performance figure and about 80% of the Cray's performance.

Yet an entire school of thought bases relative machine performance on this test (see, for example, the recent IEEE Spectrum article of Reference 5).

9. LONGER EXPRESSION (ADD, MULTIPLY, ADD, MULTIPLY)

The previous tests have been fairly simple operations chosen because they are probably widely used as components of vector codes. But in real applications, the vectorizable loops may be much more complicated, consisting of several vector operations. What is the performance of more complicated constructs? This test is designed to find out:

```
      Do 9 i = 1, N
9  X(i) = A(i)*(B(i) + C(i)*(D(i) + E(i)))
```

The test uses four operations (two multiplies and two adds) sequentially; that is, each addition and multiplication requires the results of the previous ones. The graphs in Figure 9a show MOPS as before; the MFLOPS rate is now four times the MOPS rate, since there are four floating point operations in each computation of X(i). Figure 9b has been prepared with an expanded vertical scale, since the MOPS rates were low on this test.

With the information gleaned from the previous tests, it should be possible to predict some features of the performance of this test. For example, one would expect that:

1. Scalar loop unrolling should help much less than for the previous tests, since this test includes more computation per result (and therefore a rolled scalar loop will have more instructions per loop);

2. Vector performance should be significantly better for machines capable of more vector chaining.

Quantitatively, the number of cycles per completed X(i) (i.e., the peak vector performance) should be:

Cray--capable of chaining two loads, one store, one add, and one multiply:

Cycle 1: Load D; Load E; Add

Cycle 2: Load C; Multiply: Load B; Add

Cycle 3: Load A; Multiply; Store

Thus the maximum rate should be one result in three cycles, or 39 MOPS. Since there will be some sectioning overhead, the results of Figure 9 (about 30 MOPS) are consistent.

IBM 3090--With no chaining, this machine's performance is easy to calculate; each of the following instructions requires one cycle:

Load D;
Add E;
Multiply by C;
Add B;
Multiply by A;
Store to X.

--for a maximum rate of one result in six cycles, or 9 MOPS. With some sectioning overhead, the result of about 7 MOPS is in agreement.

NAS XL--The chaining capability of this machine is intermediate between that of the Cray and the IBM 3090. The situation is actually more subtle than described above, as follows: the NAS XL's vector processor chains only within an instruction (this might be called "intra-instruction" chaining, as opposed to the Cray, which has "inter-instruction" chaining). Since NAS XL vector instructions are memory-to-memory, and the instruction set is fairly rich, each of the above tests could be implemented as a single vector instruction. Therefore the "intra-instruction" chaining usually resulted in a maximum processing rate of one complete operation per machine cycle.

The IBM 3090 also has "intra-instruction" chaining, but since its instructions are more "atomic" in nature, the above tests usually translated into two or three IBM 3090 vector instructions, with lower resulting performance.

The Cray has even more atomic instructions than the IBM 3090, but its full "inter-instruction" chaining more than makes up for the difference.

Returning to the calculation of NAS XL vector performance: Test 9 can be implemented as follows on an NAS XL, with one instruction per cycle:

Add D and E;
Multiply by C;
Add B;
Multiply by A.

--for a maximum vector rate of one result in four cycles, or 14 MOPS. Since there is no sectioning overhead on an NAS XL, that rate was actually achieved.

Test 9 therefore illustrates very important chaining properties of the machines tested. For example, one conclusion is that even though the IBM 3090's architecture (with vector registers) is much more like the Cray's than is the NAS XL architecture, the NAS XL nevertheless is much closer to the Cray's performance by virtue of more extensive use of chaining.

Test 9 also yields some important scalar information: the Cray scalar processor was able to benefit significantly from loop unrolling, contrary to expectation (whereas the IBM 3090 and NAS XL scalar processors were virtually unaffected by loop unrolling). Another way to view this result is that the Cray scalar processor's performance is much more susceptible to damage due to looping than either the IBM 3090's or the NAS XL's. As before, the Cray's scalar performance was poor for very short "vectors": it reached 50% of its peak unrolled performance at a length of about five, whereas the other two machines were above 50% of their peak performance even when only one operation was performed.

10. ARCHITECTURAL CONCLUSIONS

The above data suggest the following observations:

1. Vector performance is less related to the complexity of the operation than to the characteristics of the machine's architecture.

2. Machines with vector chaining (e.g., the Cray-XMP and the NAS XL) have substantially better vector performance than non-chaining machines (e.g., the IBM 3090).

3. The Cray's vector performance is considerably better than that of either the IBM 3090 or the NAS XL; in particular, the Cray achieves "vector break-even" at much smaller vector lengths than the other machines.

4. By contrast, the Cray's scalar processor seems quite inferior to those of the other two machines. Not only does the Cray's scalar performance suffer for small "vector lengths", but loop unrolling improves its performance even for the most complex operations. Looping seems to seriously degrade the Cray's scalar performance, compared to the other machines tested.

5. Both cache machines (the IBM 3090 and the NAS XL) degrade substantially when the data will not fit in cache. Although one might argue that vector lengths of several thousand (the size where the problem starts) are rare, the cache degradation problem may also be present for matrix operations; even modest-sized matrices may exceed several thousand elements.

6. Roughly speaking, the Cray has an impressive vector processor, and the IBM-style machines (especially the NAS XL) have impressive scalar processors. A machine incorporating the best of both might seem desirable, but in fact it might suffer from inadequate vector performance.

11. SUGGESTIONS FOR FRUITFUL VECTORIZATION

The data in this paper have shown that adequate vector lengths are mandatory in order to achieve a benefit from vector processors. The following discussion is therefore included:

11.1 VECTOR LENGTH

It is overwhelmingly important to know what vector lengths occur in actual practice. Livermore reports that they have measured their average vector length to be about 70 (Ref. 4). With such short vectors at an installation that would seem to be the preeminent vector user, one wonders what less auspicious vector lengths occur in the rest of the world. There appear to be almost no data available on this subject, although there are three reasons why vector lengths may be small in practice:

1. Many engineering and scientific applications use two- and three-dimensional matrices in their calculations. Nevertheless, many vector algorithms process only a row (or column) of a matrix at a time, so that a 70x70 matrix would generate a vector length of 70, not 4,900. And a three-dimensional 70x70x70 matrix contains 343,000 elements requiring 2.7 megabytes on an IBM-style machine, but probably still uses a vector length of only 70. Clearly not many such matrices will fit in memory; larger matrices (for longer vectors) require even more memory.

2. Livermore also reports that applications that perform physical simulations are frequently CPU-bound rather than memory-bound; for example, a three-dimensional grid of points spanning some physical system of interest. If the simulation is to be improved by, say, doubling the number of simulated points per inch (thereby increasing the required storage by a factor of eight), not only will the CPU time per complete calculation be multiplied by eight, but also the required number of time steps will be doubled. This additional doubling is caused by the fact that effects at one grid point will reach the next grid

point in half the time of the old simulation (the grid points are now half as far apart). So to simulate the same amount of physical time, the CPU time will increase by a factor not of eight, but rather of 16. Thus, paradoxically, the limit on vector length for physical simulations may not be storage, but rather CPU time.

3. Some operations on NxN matrices do not even have vector lengths as large as N. For example, matrix inversion generates upper-triangular matrices and processes them-- thus, the vector length will run uniformly from 1 up to N. Short vectors therefore appear to be unavoidable in real applications, no matter how large the matrices.

For all these reasons, vector lengths may be disappointingly small in practice (or more properly, result in disappointing performance).

Consider the data in Figures 1 through 9. Even though the Cray vector processor was always the best peak performer, at a vector length of 70 most of its advantage disappeared. For example, on the "Scalar*Vector + Vector" test the NAS XL vector processor was almost equal to the Cray's vector processor; for "Dot Product", the IBM 3090's vector processor was about as fast as the Cray's, and the NAS XL was faster; and finally, on "Add up a Vector", the IBM 3090's scalar processor was about as fast as the Cray's vector processor, and the NAS XL's scalar processor was almost twice as fast as the Cray's vector processor! (Remember that these comments are specific to a vector length of 70.)

For many applications, therefore, scalar processors provide performance as good as (or even better than) vector processors--despite a recent claim in the literature that "scalar machines have been made largely obsolete" (Ref. 5).

The results presented above suggest a number of guidelines for those who will be writing code for vector machines:

1. Above all, long vectors are required for adequate vector performance--indeed, they may be necessary to keep vector performance from being worse than scalar performance.

   a. Try for vector lengths in the hundreds or even thousands.

   b. In matrix calculations, try to vectorize a plane rather than a row. Considerable success has been reported using such techniques in special cases (Ref. 4).

   c. Use scalar operations, especially unrolled loops, where vector lengths are short. The benchmark technique reported in this paper can be used to determine what "short" means.

2. Take advantage of vector chaining where your architecture permits it.

3. For cache machines, note the limitations on vector length for good performance.

4. Avoid vector divides.

12. SUGGESTIONS FOR FUTURE VECTOR PROCESSORS

The data presented above also suggest ways in which future vector processors can achieve better performance than now available:

1. Vector processors need greatly reduced start-up times, in order to reduce the "vector break-even" lengths--which for some instructions on some machines approach the reported average vector length.

2. It appears that a vector register length of 64 (as used by Cray) limits performance. Even 128 (as used by IBM) may not be long enough. Longer vector registers are indicated.

3. Vector chaining is mandatory for good vector performance. It is especially important for register machines, since otherwise each required load and store consumes one additional machine cycle per operation. For example, the NAS XL scalar processor is frequently as fast as the IBM 3090 vector processor (a vector register machine without chaining)--yet both machines have essentially the same cycle time.

4. Cache machines may need larger caches, or some way to circumvent effects due to cache degradation.

5. Vector performance analysis tools are sorely needed. Particular emphasis should be given to reporting vector lengths, overhead involved in vector processing, and percentage of time spent executing in vector mode.

13. CONCLUSION

The benchmark technique discussed in Reference 1 has been used to produce information on vector vs. scalar performance architecture of the three machines tested: a Cray-XMP/416, an IBM 3090-200/VF, and an NAS AS/XL VIOO. The main lessons can be summarized as follows:

1. Vector performance may be worse than scalar performance for short vectors--sometimes "short" is as long as 70 elements;

2. Very long vectors (hundreds or even thousands) are required to approach the limits of vector performance;

3. The amount of vector chaining varies substantially among the available vector processors. Chaining dramatically improves vector performance by allowing several instructions to complete in every machine cycle;

4. Loop unrolling can provide significantly improved scalar performance, especially on heavily pipelined machines. The performance of unrolled loops in this benchmark is probably equivalent to that of real applications with several operations in a loop.

The data have also been analyzed to produce some simple guidelines for implementing vectorized code, and suggestions for future hardware improvements to vector processors.

CMG attendees are invited to run this benchmark free of charge, provided only that they return data to the author so that more information can be gathered on today's processors. It is not necessary to run the benchmark on a vector processor--as mentioned above, interesting data are produced even for scalar processors. At the end of the paper is a request form for a copy of the vector benchmark.

14. ACKNOWLEDGMENT

The author wishes to thank Frank McMahon of Lawrence Livermore National Laboratory, who ran the benchmarks on a Cray-XMP and supplied the Cray results that appear in this paper; suggested loop unrolling; and originated the analysis of vector length that is presented in Section 11.0 of this paper.

15. REFERENCES

1. D. S. Lindsay, "A Technique for Evaluating the Performance of Vector Processors", CMG-87 Conference Proceedings.

2. F. H. McMahon, "The Livermore FORTRAN Kernels: A Computer Test of the Numerical Performance Range", Lawrence Livermore National Laboratory Report No. UCRL-53745, Dec 86.

3. J. J. Dongarra, "Performance of Various Computers Using Standard Linear Equation Software in a FORTRAN Environment", Argonne National Laboratory Technical Memorandum No. 23, Mar 87.

4. F. H. McMahon, private communication, 1986.

5. J. Dongarra, J. L. Martin & J. Worlton, "Computer Benchmarking: Paths and Pitfalls", IEEE Spectrum, pp. 38 43, July 1987.

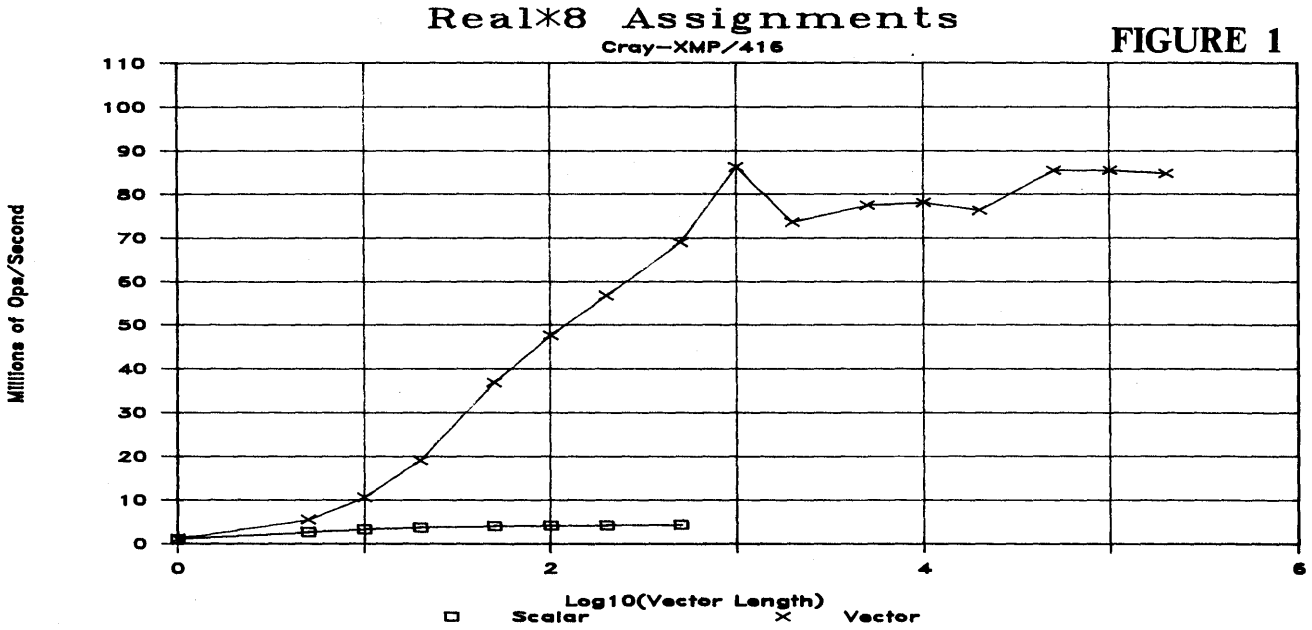
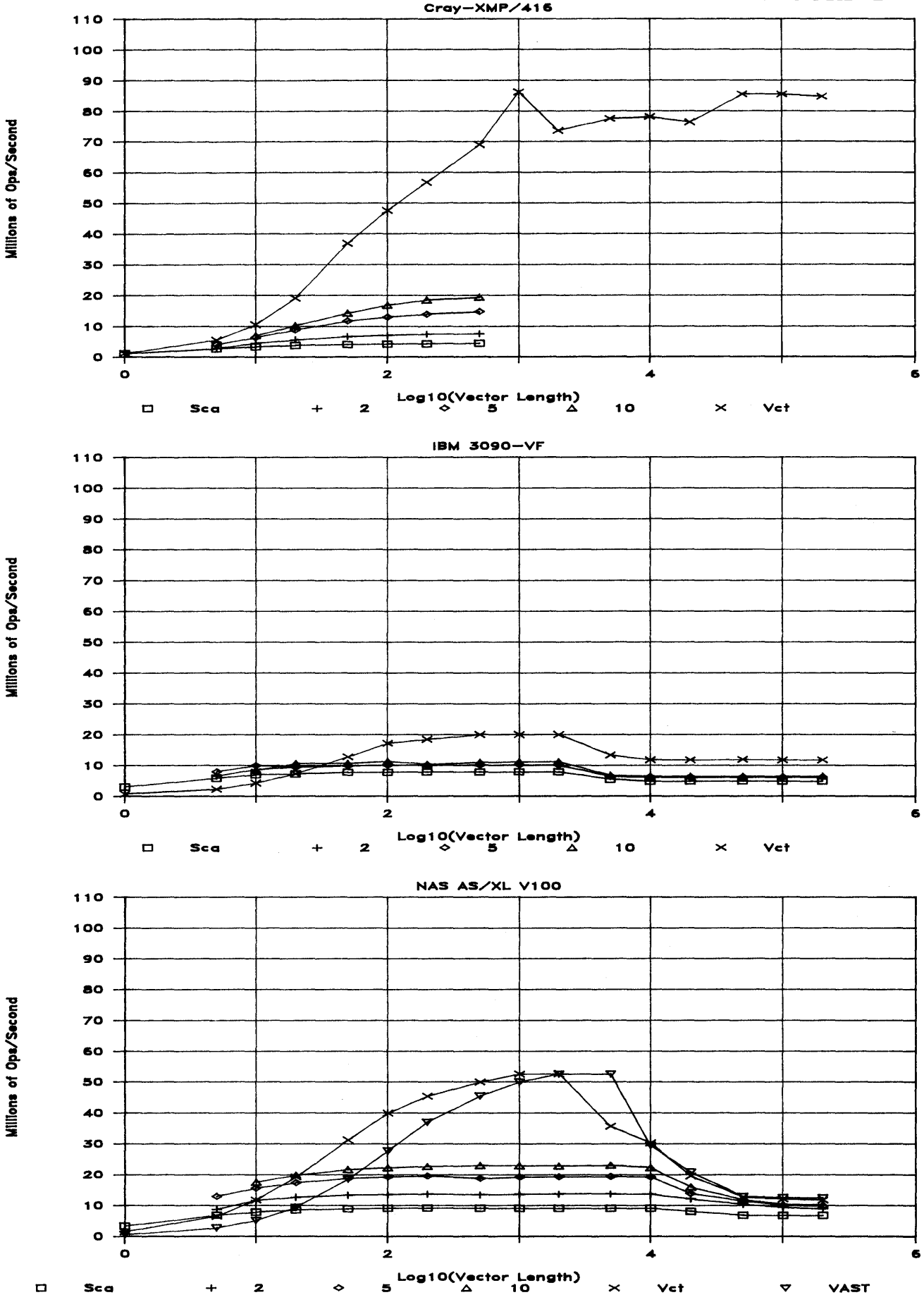


FIGURE 1

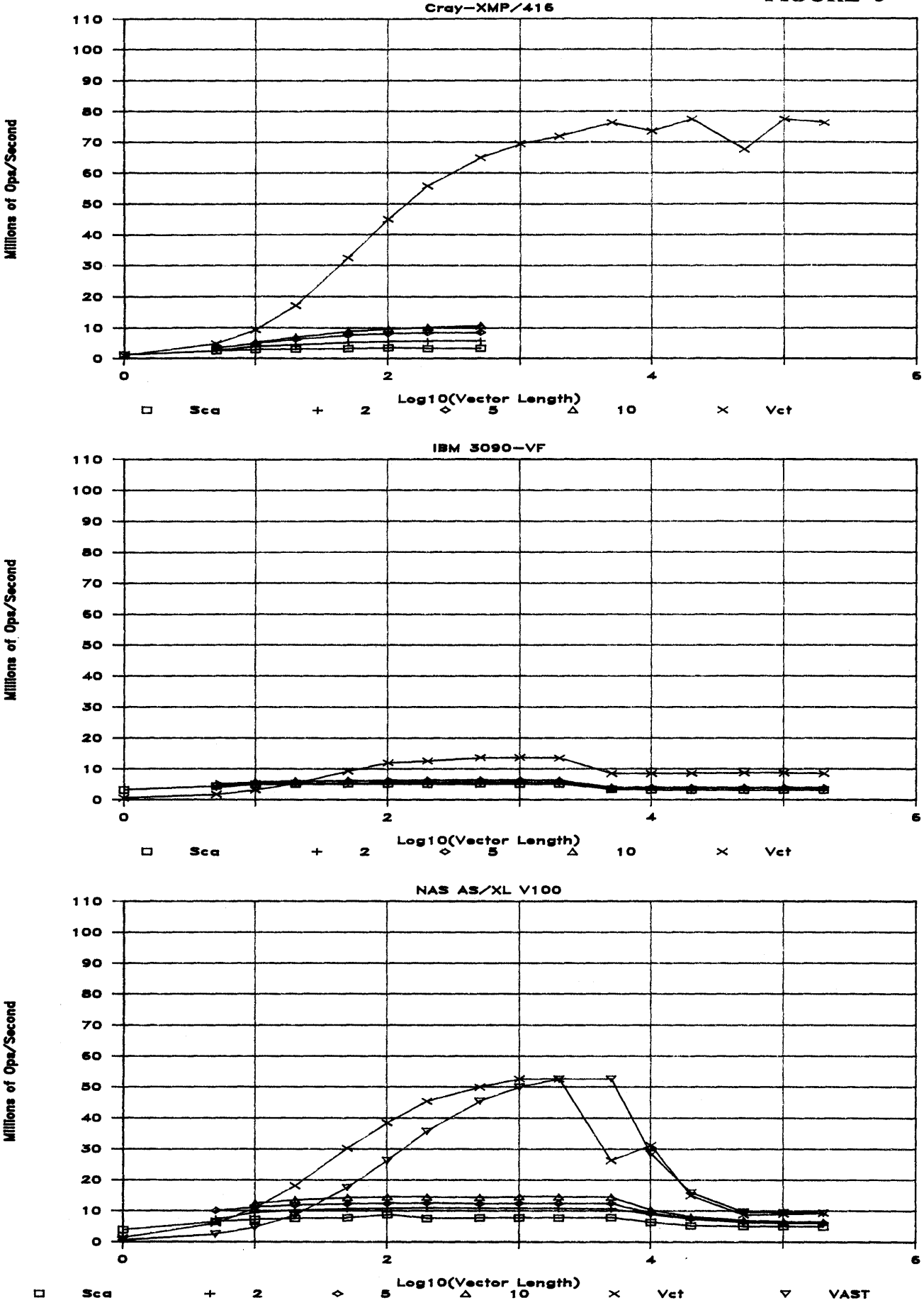
Real*8 Assignments

FIGURE 2



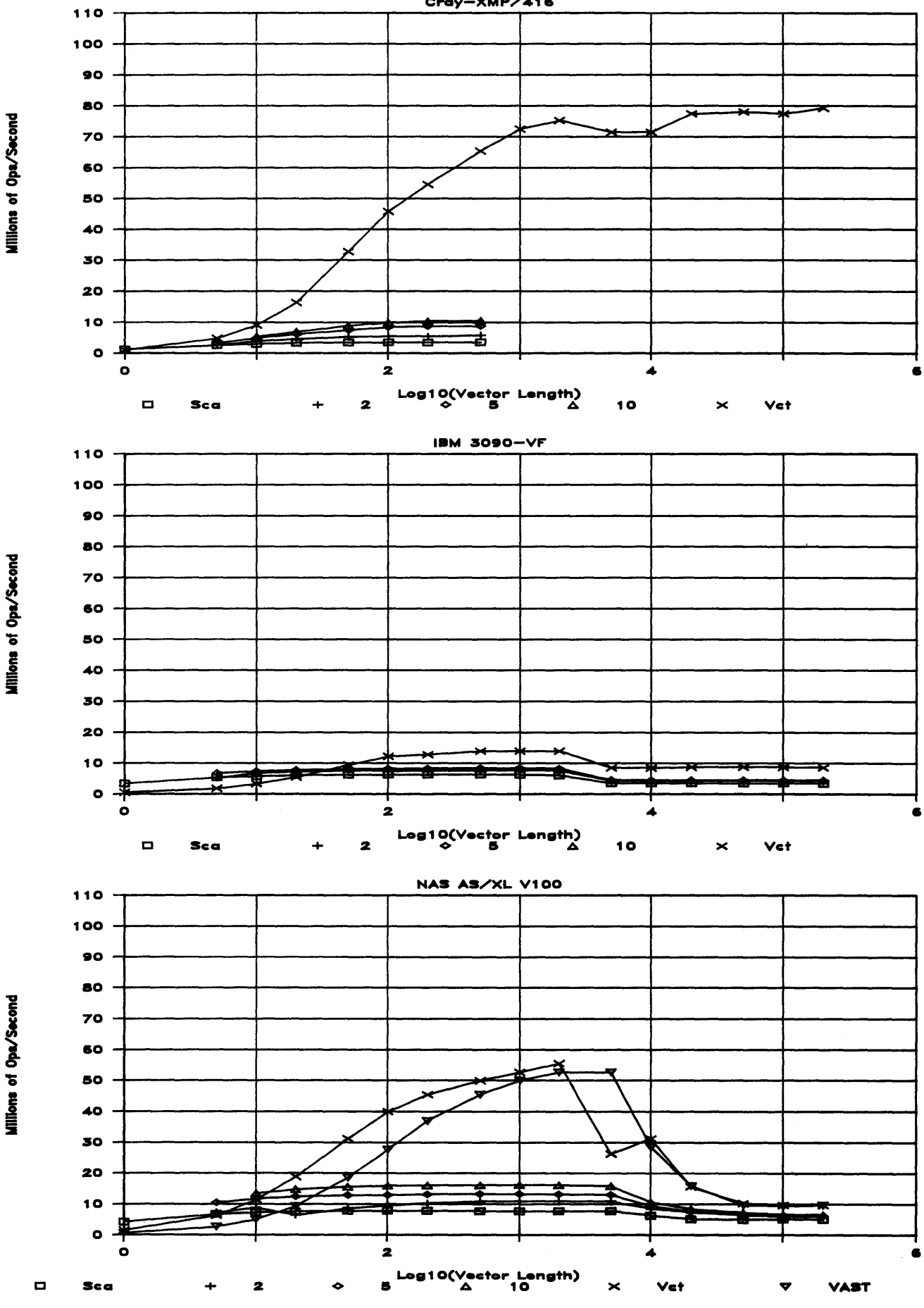
Vector Multiply

FIGURE 3



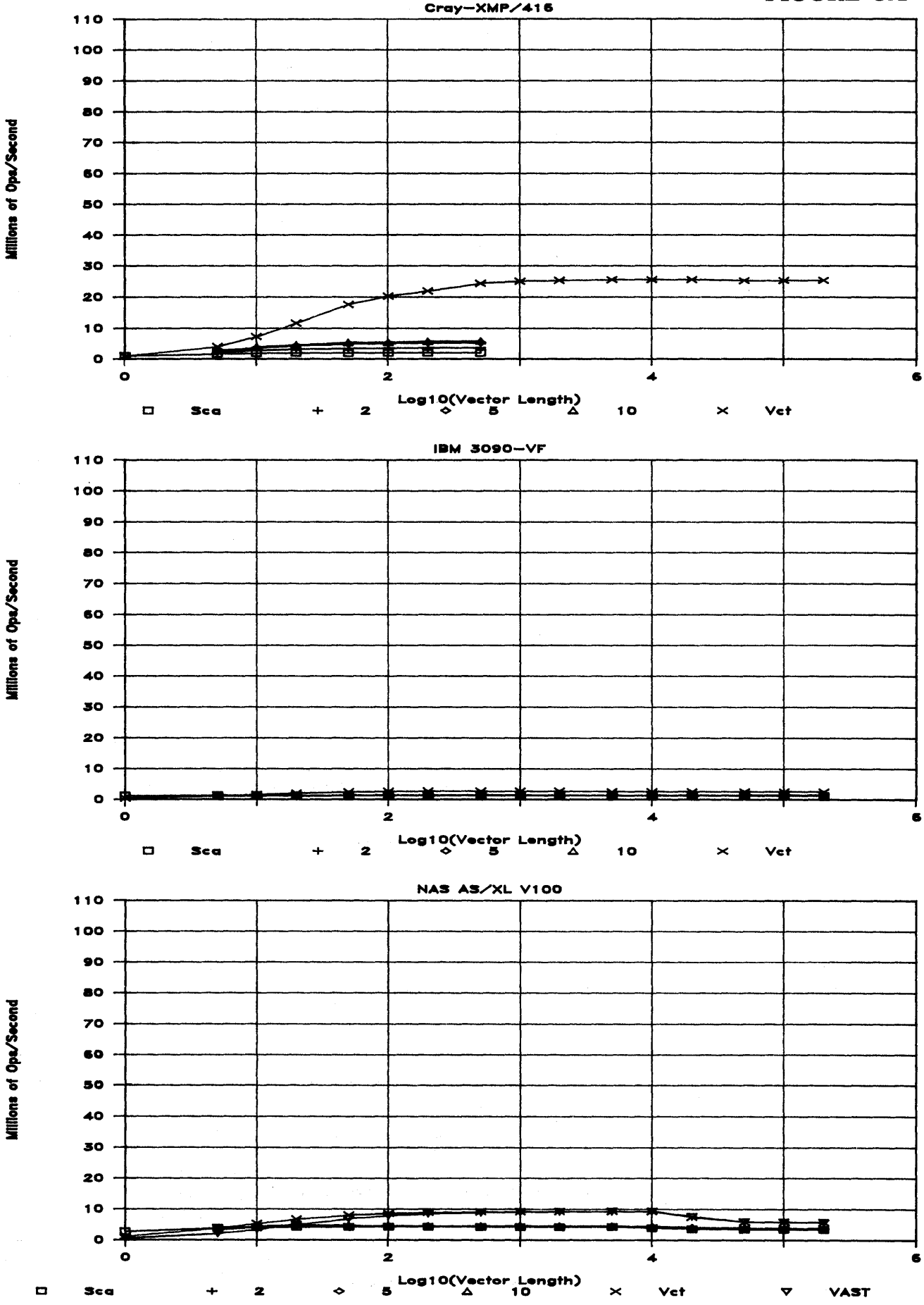
Vector Add

FIGURE 4



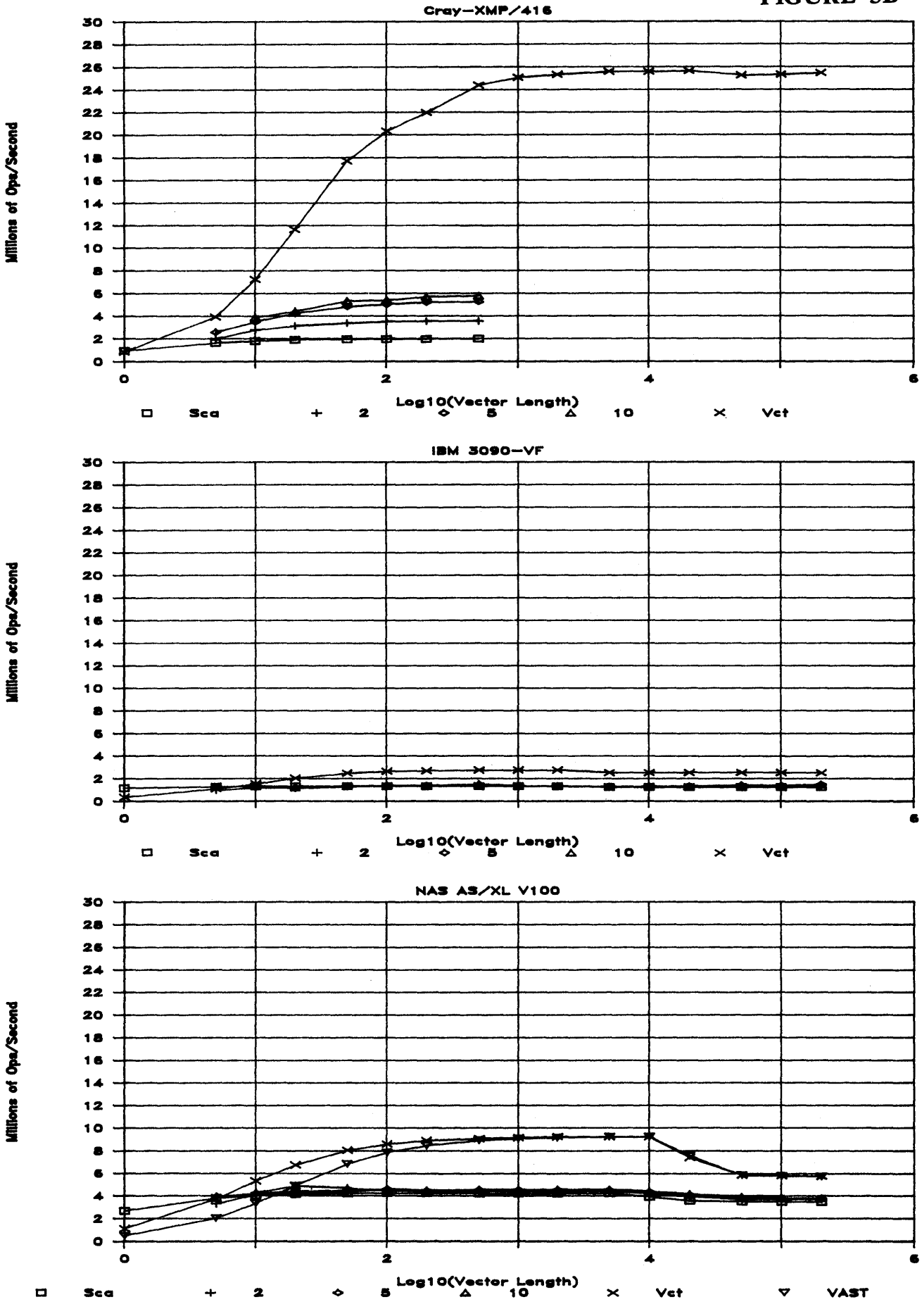
Vector Divide

FIGURE 5A



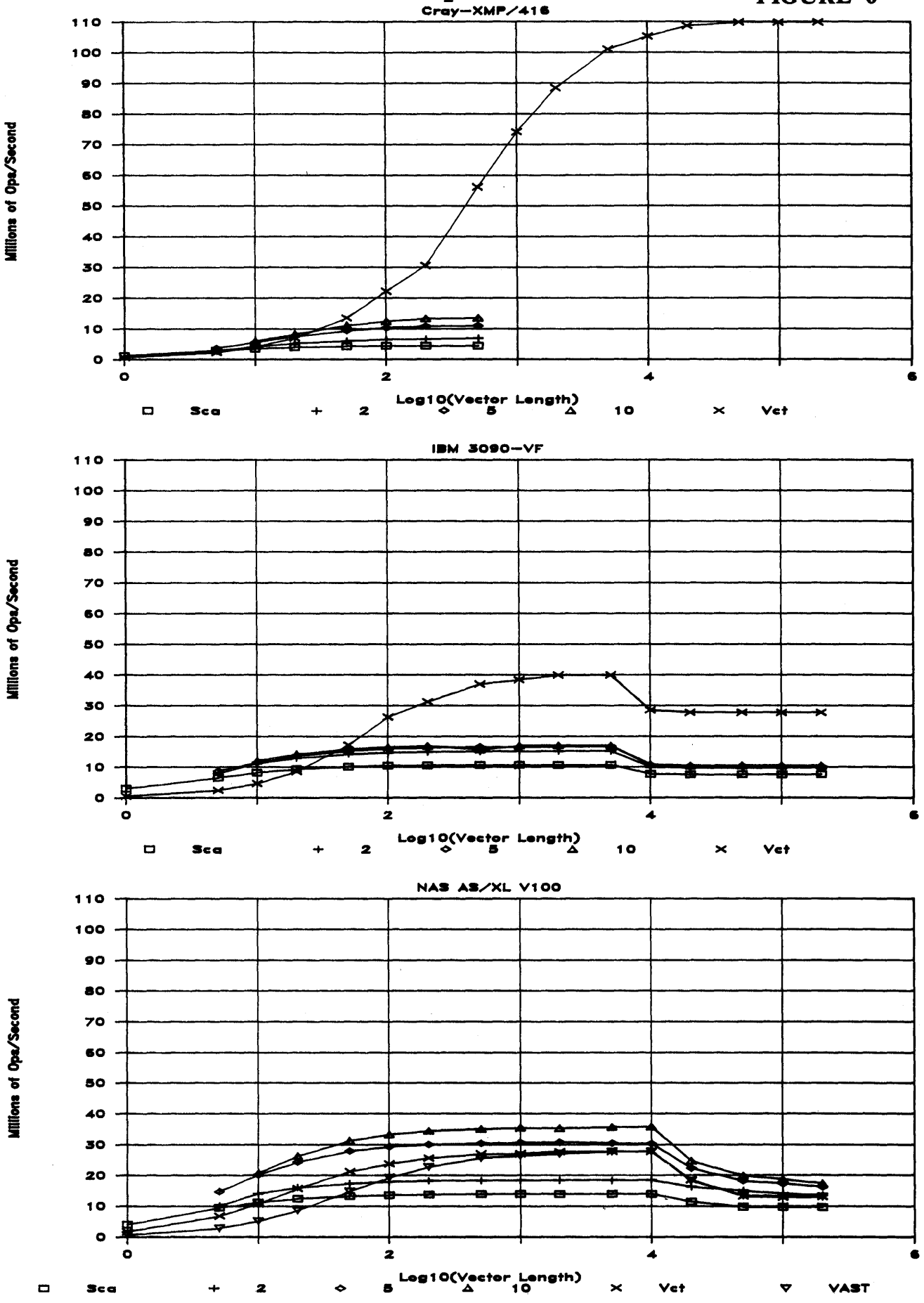
Vector Divide

FIGURE 5B



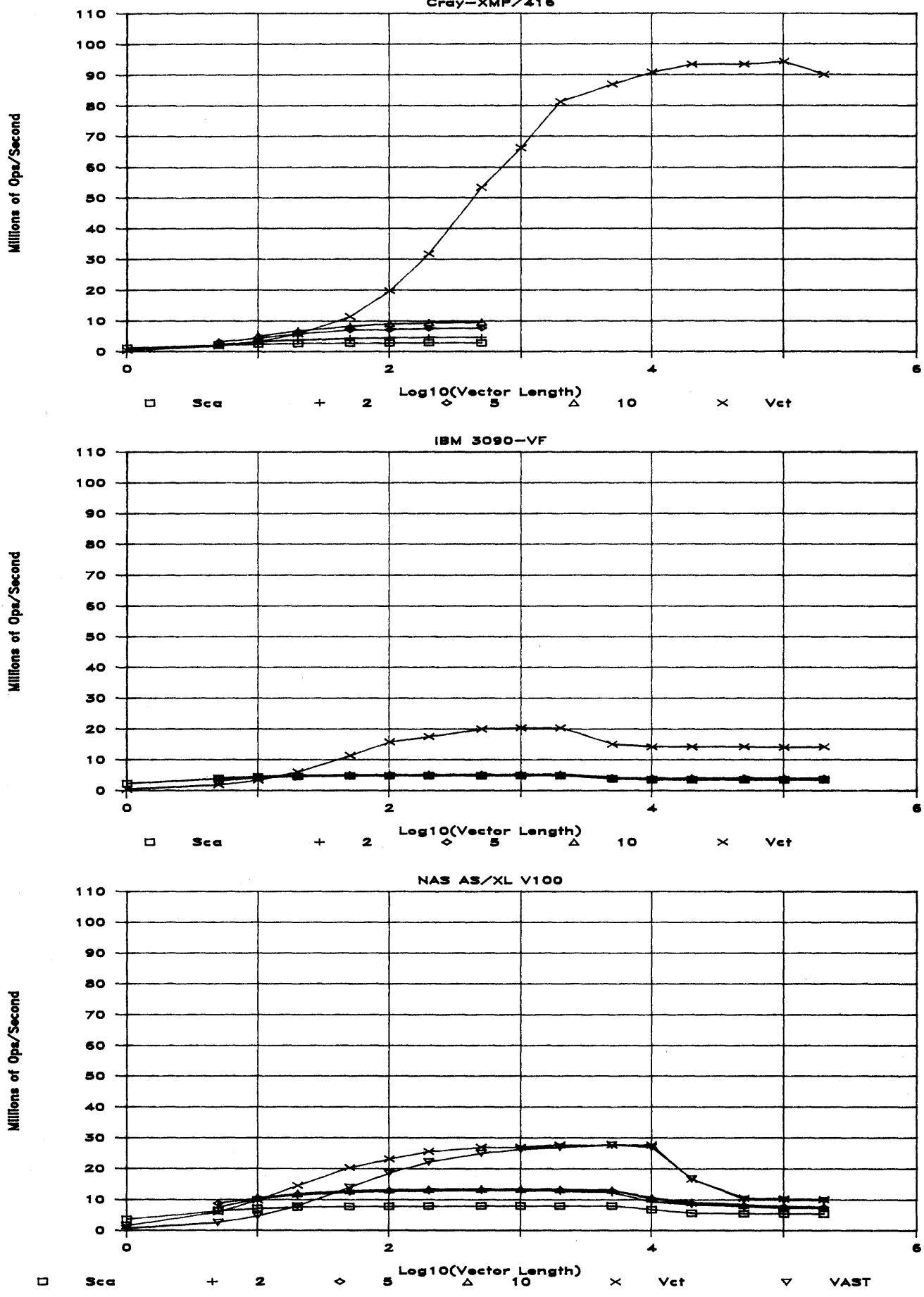
Add up a Vector

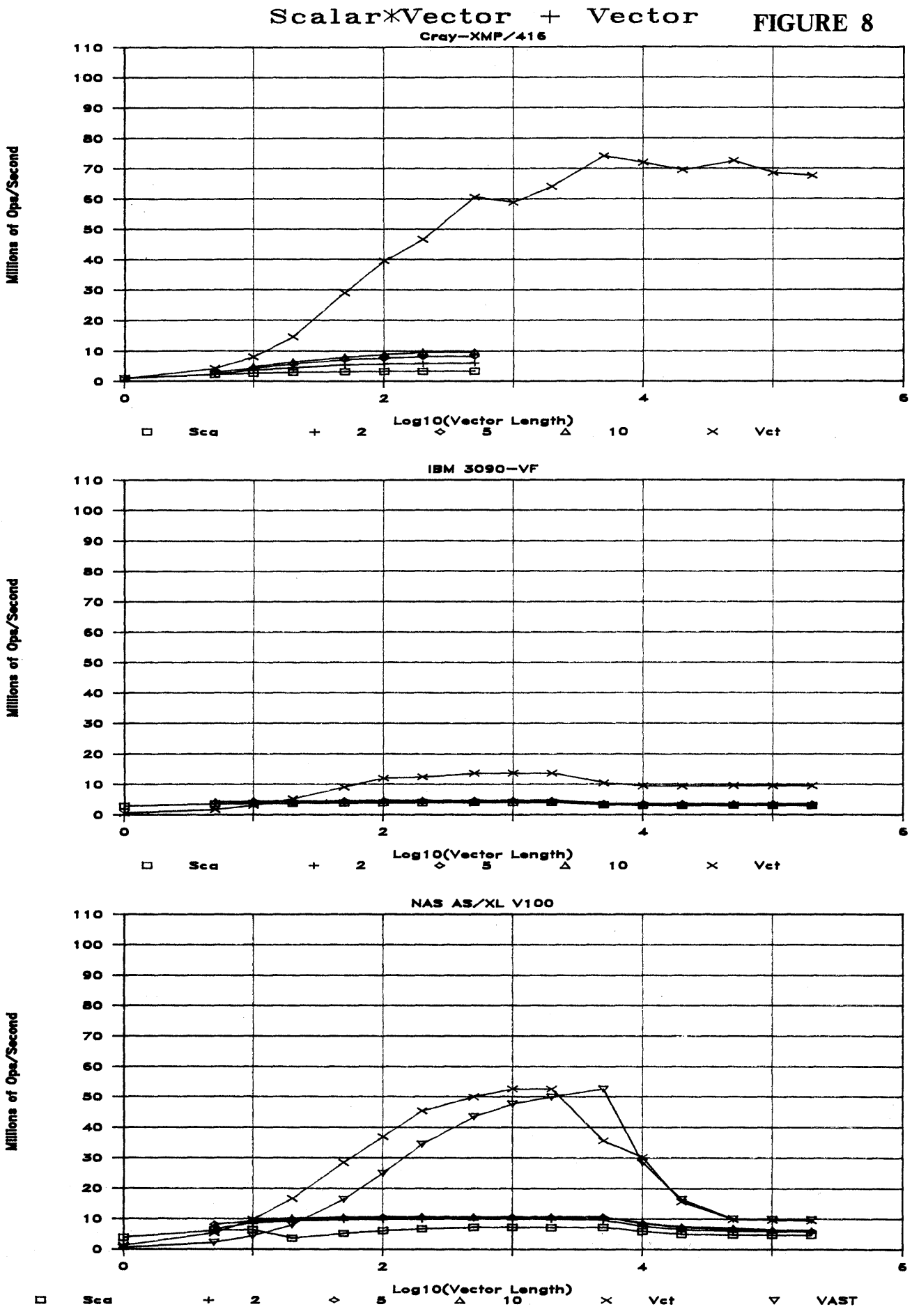
FIGURE 6



Dot Product

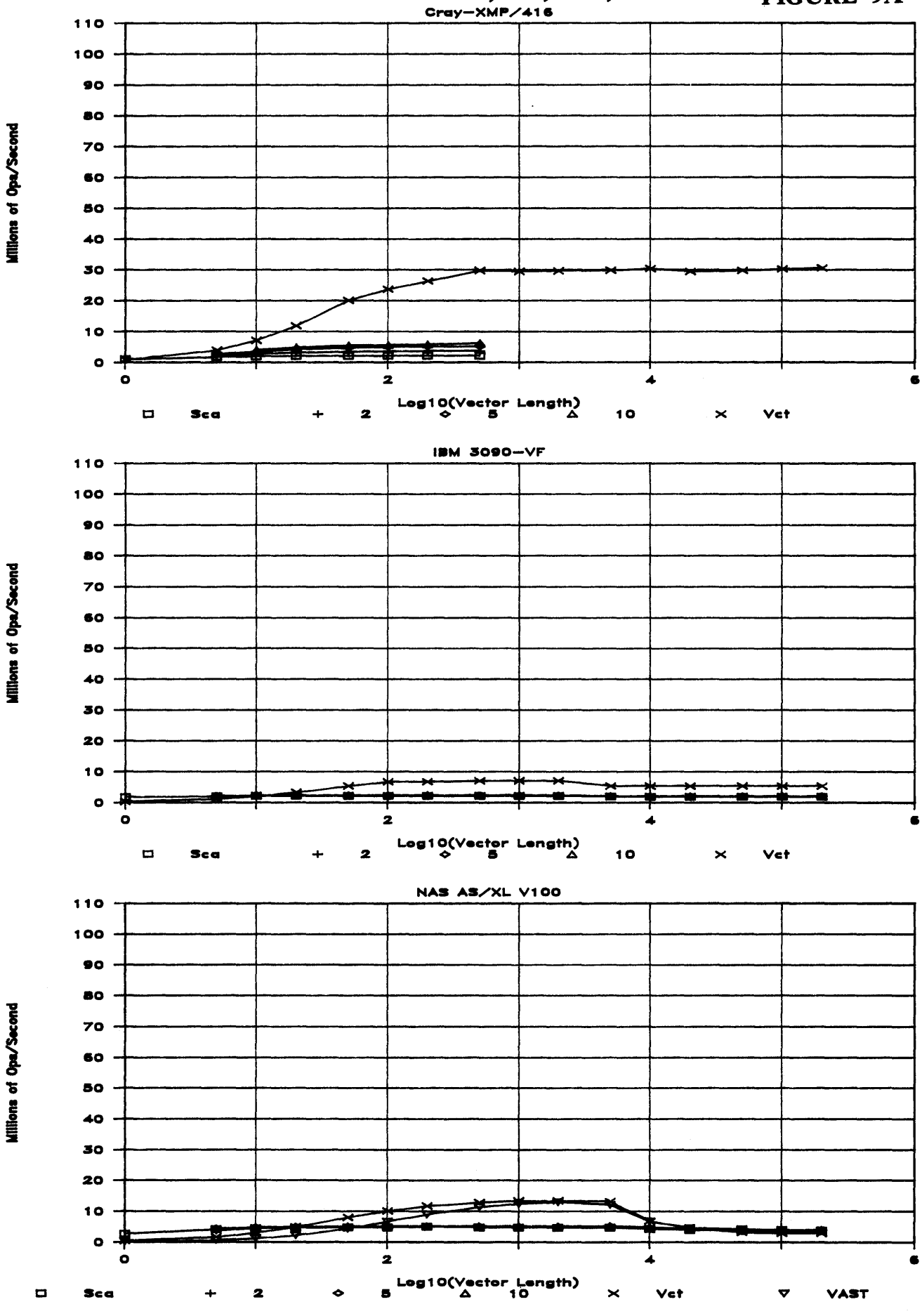
FIGURE 7




Scalar*Vector + Vector
FIGURE 8
Cray-XMP/416
IBM 3090-VF
NAS AS/XL V100
Millions of Ops/Second
Log10(Vector Length)
Sca
2
5
10
Vct
VAST

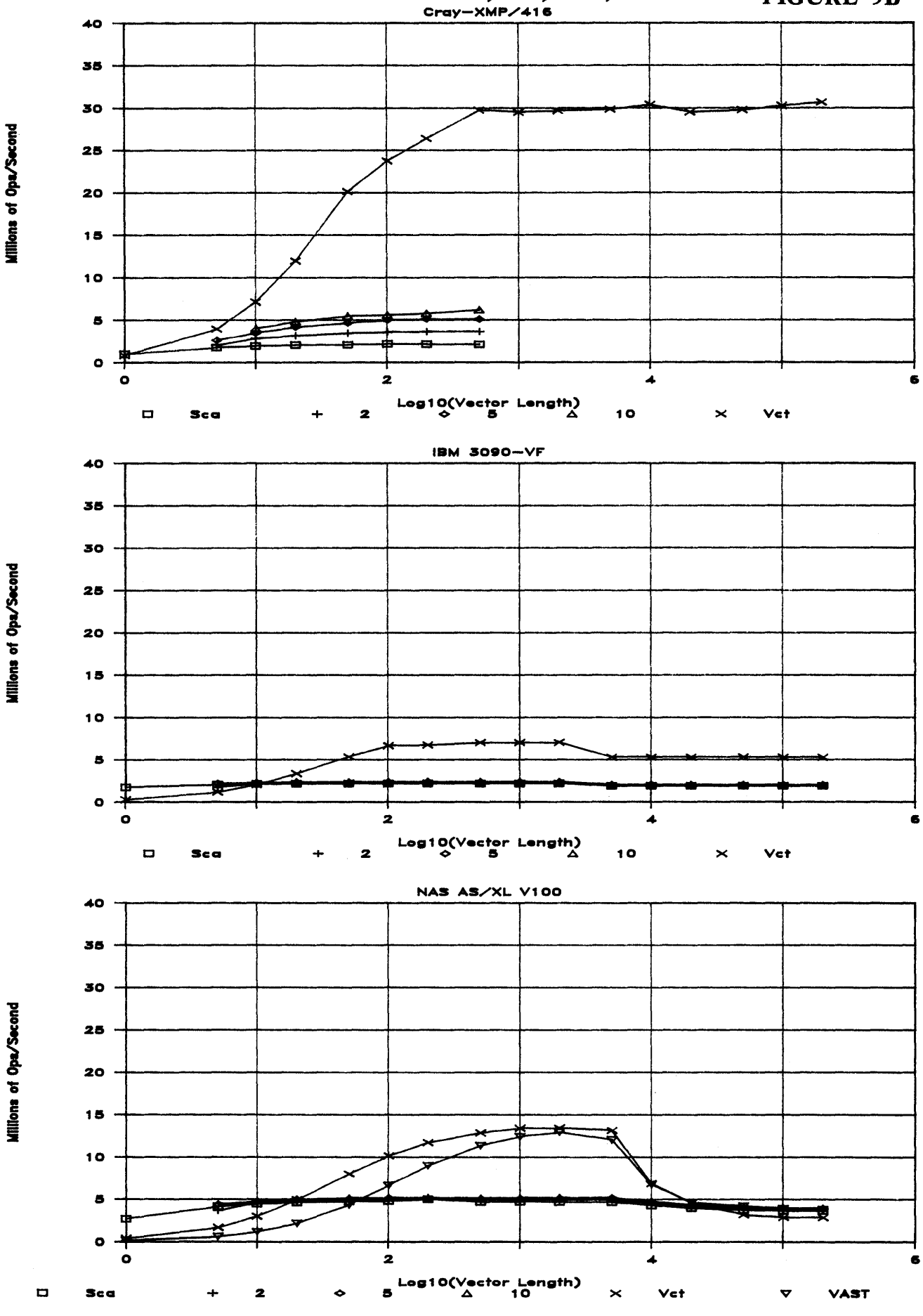
Vector +, *, +, *

FIGURE 9A



Vector +, *, +, *

FIGURE 9B



BENCHMARK REQUEST FORM

Name _____

Company _____

Address _____

City/State/Zip _____

Telephone (___)_____

I was an attendee at CMG-87. I enclose a 51/4" floppy diskette for a copy of (1) the FORTRAN source for the Vector Architecture Benchmark, (2) the input file, and (3) instructions.

In return for these I agree to provide within 30 days:

1. Execution times for at least one computer. NOTE: It is not necessary to run this benchmark on a vector processor;

2. Information about test conditions including (at least):
   a. Whether stand-alone or normal production run;
   b. Job priority;
   c. Machine model and characteristics (e.g.,memory);
   d. Compiler and version;
   e. Whether vector or scalar. For scalar, unrolling level;
3. Listing of timer routine used if not already in benchmark source;
4. Permission to report these data.

I understand that future versions of this Vector Architecture Benchmark may include the names of individuals who fail to meet this agreement, and I would welcome such public exposure should I fail to provide the agreed materials within 30 days.

_____
Signature

_____
Date

Mail to:

Dr. David S. Lindsay
National Advanced Systems
Mail Stop 32/75
PO Box 54996
Santa Clara, CA 95054
(408) 970-1030

A TECHNIQUE FOR EVALUATING THE PERFORMANCE OF VECTOR PROCESSORS

Dr. David S. Lindsay
National Advanced Systems

ABSTRACT

With the advent of vector processors on general-purpose machines such as the IBM 3090, vector processing has emerged from the scientific arena and become a topic of interest to large systems users in general. Performance analysts at traditionally "general purpose" sites have begun receiving requests to analyze the performance of such processors.

This paper presents a technique for investigating the performance of vector processors, a technique that yields such vital information as the vector break-even length, the ability of the processor to chain, and, for cache machines, the performance implications of referencing data that are not in cache. The technique also yields much information about the underlying architecture, giving rise to a benchmark called the "Vector Architecture Benchmark". A companion paper (Ref. 1) presents results of this benchmark on some of the most widely used vector processors.

1. INTRODUCTION

Analyzing vector processing is a "natural" for performance analysts, since there is only one reason for vectorization: performance. A CPU gains no new functionality by acquiring a vector processor; anything that can be done by a vector processor can easily be done by a scalar processor--more easily, in fact, since traditional methods and programming languages can be used in scalar mode. The only justification for an expensive vector processor is speed.

Unfortunately, the performance of vector processors is more fragile than that of general purpose processors: dramatic (factor of 10) speed improvements are possible, but vectorized programs can also run slower than equivalent scalar programs. Performance immediately becomes an issue when a data center is asked to justify the acquisition of a vector processor. Thus, it is essential for CPE professionals in vector installations to become cognizant of the deficiencies and opportunities inherent in vector processing.

Before exploring methods of evaluating the performance of vector processors, it is important to understand how vector processing works and under what circumstances good and bad performance should be expected.

1.1 OVERVIEW OF VECTOR PROCESSING

"Vector processing" refers to highly parallel processing. Vector processing includes the following characteristics:

1. Vector processors are usually designed for Engineering/Scientific" applications.

2. Special vector hardware is supplied with the CPU.

3. Special machine instructions are required to use the vector hardware. These in turn must be generated somehow--by the user writing his own programs in assembler language, by a special vectorizing feature of a high-level language compiler (usually FORTRAN), or by embedding (into regular source code) calls to a library of specially vectorized subroutines.

4. Vector processors usually execute floating point (i.e., the FORTRAN REAL data type), although many machines also have integer and logical vector capabilities.

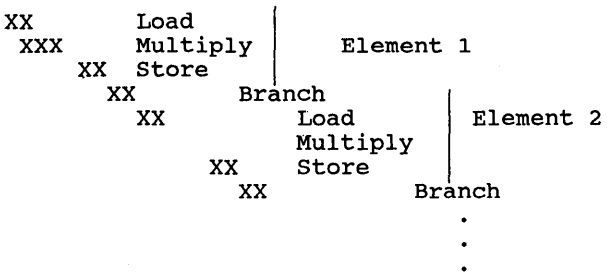
5. Vector instructions are usually simple instructions such as multiply, add, compare, etc.; complex instructions such as the COBOL "Edit" instruction are not available in vector hardware, and flow-of-control instructions such as branches are by their nature not vectorizable.

6. If the CPU in question has a memory cache, the vector instructions may or may not reference it (a cache is a small, high-speed auxiliary memory that is used to store the currently referenced code and data of a program. A cache is invisible to the programmer, being maintained by the CPU itself). Not all vector processors have caches; a Cray-XMP does not, whereas both an IBM 3090 and an NAS AS/XL do. If the vector instructions reference cache, vector performance may degrade if the associated data are not in cache. Possible reasons for data not being in cache are: (1) the application program has poor "locality of reference"; (2) many program interrupts have overwritten the cache; or (3) the vectors are too large to fit in the cache. CPU caches may therefore affect vector performance.

As an example of vectorization, consider the following FORTRAN DO-loop:

```
      DO 1 i = 1, N
1     X(i) = Y(i)*Z(i)
```

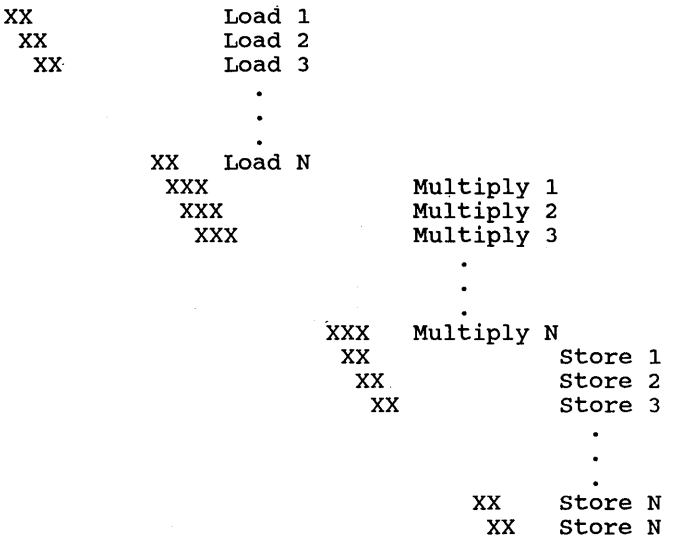
Where N is called the "vector length". On a scalar (i.e., non-vector) machine, the execution might proceed as shown in Figure 1, where the X's are meant to represent individual machine cycles consumed by the associated instruction. The number of X's is purely illustrative--whether multiplies actually take three cycles, and loads and stores two, is irrelevant to the current discussion. Those values are used in order to be specific about the timing computations. In the example of Figure 1, each instruction must complete before the next can start (e.g., you can't store a result that has not yet been generated); thus, the loop is doomed to consume several machine cycles (nine in this case) per multiply.



Timeline for Scalar Multiplies in a Loop
(Requires 9N cycles)

Figure 1

Using a vector processor, however, the same loop could be processed as depicted in Figure 2. The importance of Figure 2 is that the individual operations (load, multiply, and store) are each pipelined so that, as shown, a new operation can be started in each machine cycle. There is also some overhead involved in starting the vector instruction (e.g., the programmer or compiler must set certain values and pointers in registers, and the hardware may also take time to decode the vector instruction and start the vector processor).



Timeline for Vector Multiply
(Requires 3N Cycles + Start-up Time)

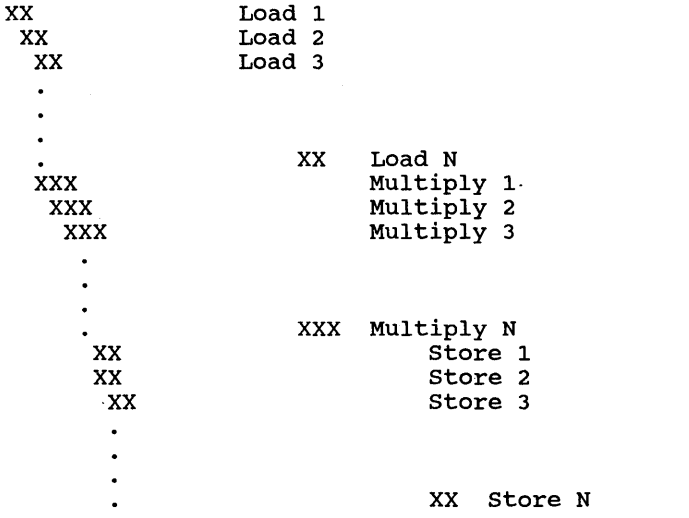
Figure 2

The number of repetitions of the basic instruction (in this case, the number of multiplies) is called the "vector length", N in this case. From Figure 2, one can easily see what makes a vector processor so attractive--in this example, the processing time is only 3N cycles plus the start-up time. Note that the value 3N is not dependent on the number of cycles required for a multiply or a load; it is a consequence of the pipelining of the three vector instructions (load, multiply, and store). The IBM 3090-VF is an example of a vector processor that operates as shown in Figure 2.

Therefore, if the vector length N is large enough to offset the start-up time, vector processing will be faster than scalar processing.

1.2 VECTOR CHAINING

There is another level of sophistication that can be implemented, as shown in Figure 3. This additional level is called "Vector Chaining", or just "chaining" for short. Chaining is possible because the first multiply could logically be started the moment the first load is complete--without waiting for the rest of the loads to complete, or even start. Similarly, the first store can begin as soon as the first multiply is complete. Figure 3 portrays these timing relations--the dotted lines indicate the conclusions of the first operations of each vector instruction.



Timeline for Vector Multiply With Chaining
(Requires N Cycles + Start-up Time)

Figure 3

Thus, if nothing interrupts the various pipelines, the loads, multiplies, and stores all proceed in parallel; the result is N multiplies in N cycles plus start-up time.

The hallmark of vector chaining is thus one full operation per machine cycle. This speed is independent of the actual number of cycles required for (say) a multiply; the point is that the load, multiply, and store are all overlapped and pipelined. The architecture permits one operation per cycle, regardless of the length of an individual operation.

Vector chaining is obviously a very powerful feature, since it means that many operations can be performed simultaneously. For example, a Cray-XMP is capable of chaining two loads, one store, one multiply, and one add (in addition to various other operations). This means that, once the vector units have been started, a Cray can execute those five operations in each machine cycle, potentially executing at many times the MIPS rate implied by its cycle time.

To summarize this overview of vector processing, it is helpful to refer again to the example of the vector multiply:

1. In scalar mode (which is what a normal programmer uses all the time), the execution proceeds in a loop, roughly as depicted in Figure 1. Using the timing example presented there, each multiply would consume nine machine cycles (a load, a multiply, a store, and a branch). Little or no overlap of successive instructions is possible.

2. Using a vector processor that is not capable of chaining, execution proceeds as shown in Figure 2; the execution time will be a constant start-up time dependent on the program and the processor (but not on the data or the vector length),

followed by three cycles per multiply.

3. Using a vector processor that is capable of chaining, the processing time will be some other start-up time, again independent of the data or the vector length, followed by only N machine cycles.

1.3 PERFORMANCE DEFICIENCIES OF VECTOR PROCESSING

The above discussion has noted that some "start-up time" is consumed by a vector processor. This time is potentially disastrous from a performance viewpoint: if the vector length is not long enough to compensate for the start-up time, the vector instruction will be slower than the analogous scalar loop, regardless of how fancy (and expensive) the vector hardware is. This problem is real: the companion paper (Ref. 1) includes many examples where a vector length of 20 to 50 is required before a vector processor achieves the same performance as the scalar processor. Longer vectors are required to exceed scalar speed.

One can see that the performance of a vector processor is more fragile than that of a standard processor. In addition to all the usual performance pitfalls, vector processors are subject to:

1. Poor performance on short vectors.

2. Cache machines may degrade for long vectors if the cache size is exceeded. Such cache degradation may be more pronounced in vector mode than in scalar mode, because the vector speed is generally faster than the scalar speed.

3. If vector chaining is not implemented in the hardware, the vector/scalar performance ratio will decrease as the complexity of the vectorized expression increases; but if chaining is present,then the vector/scalar performance ratio can improve with increasing complexity.

The analysis approach described below allows a CPE professional to gain insight into how these and other effects influence the vector performance of a machine.

2. PREVIOUS BENCHMARK TECHNIQUES

Apart from some early work designed to determine the details of cycle-by-cycle machine execution (Ref. 2), most studies have ignored the details of vector processing. For example, the following techniques have recently been published:

1. Run a job, get a CPU time (Ref. 3).

This is a particularly uninformative test, since many important effects are ignored:

* The percentage of vectorized code is unknown.

* The vector lengths of the data are unknown.

* No insight is gained into vector performance--this technique is a pure "black box" with no windows. One can conclude at most that such-and-such a program took some time in scalar mode, another in "vector mode"--but can the compiler be improved ? Would slightly different data (with perhaps different vector lengths) yield drastically different performance? What about performance on other programs? Almost nothing of general applicability is learned from such a test.

2. Run a standard, vectorized package such as LINPACK using various vector lengths (Ref. 4).

This technique is marginally superior to the first technique, since something is being varied from one measurement to the next; but what? In the case of LINPACK, one actually sets a matrix size. What relation does this have to vector length? Only those intimately familiar with the package would know.

Another property of LINPACK is that it executes only a tiny subset of the available vector instructions--in the case of the vectorized LINPACK code that was run at NAS, there were only three different vector op-codes in the entire source. Such a small subset of the available instructions could easily yield a misleading idea of the vector performance of a machine.

Also, as was the case with method 1, the percentage of vectorized code remains a mystery. One might assume that it is 100% for a mathematics package like LINPACK, but is it really?

3. Run kernels from longer jobs, e.g., the Livermore Kernels (Ref. 5).

This technique is a considerable improvement over the previous ones. The Livermore Kernels contain 24 code fragments, each at three different vector lengths, every one of which is separately timed; the code in each timing loop is known, and the vector lengths are roughly stated in the tests' output. This benchmark is likely to yield a good idea of the spread in performance that real applications will exhibit--but it does not tell why.

A significant shortcoming of all of the techniques listed above is that they give no indication of the potential for improvement. Would performance improve if the vector lengths were increased? If so, how much? What vector lengths should new applications have for optimal performance? How about vectorization--was each loop fully vectorized? Might the next version of the compiler provide better performance? Is the current code optimal, so that nothing better can be obtained? Are some vector instructions much faster than others, so that certain instructions are to be preferred or avoided?

Because of the significant deficiencies inherent in existing techniques, a different philosophy was adopted for a vector benchmark, one that yields insights into the reasons for performance benefits and deficiencies. The technique is an outgrowth of earlier work on scalar machines (Ref. 6).

3. REQUIREMENTS FOR AN INFORMATIVE VECTOR BENCHMARK

Early in the study of vector performance at NAS, it became clear that there were significant issues that were either unique to vector performance or dramatically more important for vector code than for scalar code:

1. Vector speed vs. vector length;

2. Vector vs. scalar performance;

3. Sensitivity to data caching;

4. Ability to chain.

This list is not exhaustive, but it represents a necessary first step. Without a clear grasp of these issues, a CPE analyst faced with a vector processor is merely fumbling in the dark.

As mentioned above, an important parameter for a vector processor is the "vector break-even length". The approach described below measures that length (as well as many other important parameters) for a variety of vector instructions. The approach taken for the vector benchmark was:

1. Measure CPU time for several "atomic" vector instructions (such as "add", "move", "multiply", etc.).

2. Use vector lengths from 1 to 200,000, thereby (hopefully) spanning the entire range of sizes of most machines.

3. Insure 100% vectorization by including only a vector operation (or its analogous scalar loop) in the timing data. With the results of this test, one can easily compute the performance of partial vectorization. But if one only knows that "some" of the code was vectorized, of what use is the resulting performance figure?

4. Minimize measurement overhead and variability by carefully designing the timing technique.

In order to eliminate CPU measurement overhead from the timing data, the following technique (suggested by Dr. Tom Bell) was used:

```
Get CPU time;
      Execute 20 identical vector
      operations;
Get CPU time;
      Execute 40 identical vector
      operations;
Get CPU time;
```

By taking the difference between the two resulting time intervals, one neatly subtracts the overhead involved in calling the CPU clock, thereby yielding a nearly pure CPU time for 20 vector operations. In addition, other common elements in the code also cancel (so that, for example, the vector operations can be placed in subroutines for improved code modularity and readability without affecting the resulting timings).

Unfortunately, a problem arises for short vectors: the CPU times may be so small that the system's CPU timer does not give adequate resolution. That problem can be solved by looping, as follows:

```
Get CPU time;
      Do several times:
        Execute 20 identical vector
        operations;
Get CPU time;
      Do several times:
        Execute 40 identical vector
        operations;
Get CPU time;
```

Because the method of subtracting the two time intervals eliminates common factors, the additional overhead for setting up the "several times" loops is not measured. Only the branch and loop control times appear in the resulting measurement, and they will be both (a) small in comparison with the body of the loops, and (b) the same whether vector or scalar code is used for the "vector operations", so that the vector break-even point will not be affected. (One must of course remember to divide the resulting CPU times by the value of "several" that is used in the loop.)

The effects of cache can be determined by inspecting the timing data for various vector lengths. Because the vector operations are repeated, the data will be in cache as soon as the first pass is complete. In fact, the operations are usually repeated thousands of times; therefore, the resulting time per vector operation will be almost uncontaminated by cache misses, so the timings will refer to 100% in-cache data. But when the vector length is long enough, the data will not fit in cache, and the times may suddenly reflect 100% cache misses. Thus, a plot of MIPS rate versus vector length reveals the effect (if any) of cache misses (Ref. 1), and will give an indication of what can be expected if a real program executes in or out of cache.

The data type REAL*8 (i.e., double precision floating point) was selected for use throughout the benchmark, since most vector users report that double precision is required for their applications. The Cray-XMP, one of the machines for which data are reported in the companion paper, has 64 bit words, and REAL*8 is its natural mode; Cray users refer to 32-bit floating point as "half precision".

The individual vector operations that are timed in the vector benchmark are listed below. The FORTRAN code comprising the test loop is shown in each case; this code was compiled for the scalar tests, and passed through pre-processors, vectorizing compilers, or used as a model for assembler hand-coding in the vector cases.

These particular tests were chosen because of their "atomic" nature. Also, they include the commonly used operations--most calculations will be composed of sequences of these instructions:

1. Vector Assignment

```
        Do 100 i = 1, N
100     X(i) = Y(i)
```

This test was included since the only arithmetic it performs is for looping. Thus any performance effects due to floating point arithmetic speed will not be present. The test therefore shows the "raw" speed of the vector processor--an essential performance figure for adequate understanding.

2. Vector Multiply

```
        Do 200 i = 1, N
200     X(i) = Y(i) * Z(i)
```

In machines with chaining, the multiply instruction should execute at the same rate as the assignment instruction of Test 1, apart from a potentially different start-up time. Comparing the two instruction rates on long vectors will therefore show whether the vector processor in question is capable of chaining.

3. Vector Add

```
        Do 300 i = 1, N
300     X(i) = Y(i) + Z(i)
```

In a properly designed vector machine, both multiply and add should take exactly the same time, apart from start-up effects. Some readers might be surprised at that statement; but consideration of Figures 2 and 3 above show that the time for vector processing is independent of the operation performed, so long as the operation can be pipelined. Test 3 was included to demonstrate that feature of vector processing.

4. Vector Divide

```
        Do 400 i = 1, N
400     X(i) = Y(i) / Z(i)
```

Having just stated that most vector instructions execute at the same rate, it is important to realize that divide is an exception. This benchmark has encountered vector divide instructions ranging from slightly under five cycles per divide (Cray-XMP) to 20 cycles per divide (IBM 3090-VF) (Ref. 1). The problem with divide appears to be that the instruction cannot be pipelined.

By inspecting the instruction rate of the divide test, one can determine how much of a performance penalty is incurred by divide operations.

5. Vector Summation

```
        SUM = 0
        Do 500 i = 1, N
500     SUM = SUM + X(i)
```

Summation (adding up the elements of an array, rather than adding two arrays together) is very useful as a component of the vector "dot product" operation and in other applications. Naively, one would expect that "summation" could not be vectorized: in an expression like X(l) + X(2) + X(3), the addition of X(3) cannot start until there is something to add it to--until the first add has completed.

This naive approach is both right and wrong: if one views the summation as specifying the order of the additions, vectorization is not possible. But if reordering is permissible, then the summation can be divided into two chains, as follows:

```
X(1) +    X(3)  +      X(5)   +     . . .
     X(2) +     X(4)    +      X(6)  . . .
```

If this reordering of the additions is acceptable (which it may not be, since the result of a series of floating point adds will in general be slightly different if the order of summation is changed--round-off errors may accumulate differently), the chains can be alternated in the adder, and it is no longer the case that each add must complete before the next can start.

When both chains are complete, two numbers remain; a single scalar addition then completes the summation. This is the technique of "Partial Sums", and is used on the Cray-XMP and the IBM 3090-VF. Actually, the summation is usually divided into more than two independent chains. For example, the IBM 3090-VF uses four chains, and the Cray-XMP uses eight.

Inspection of the timing data from Test 5 will show whether a "partial sum" technique is being used, since an execution rate of one add per machine cycle is generally not possible without partial sums.

6. Dot Product

```
        DOTPROD = 0
        Do 600 i = 1, N
600     DOTPROD = DOTPROD + X(i)*Y(i)
```

The "dot product" operation (more formally called "inner product") is very important in physics and engineering problems; a popular belief is that dot product is the most common vectorized construct.

Since dot product necessarily uses the summation function, it is interesting to compare the speeds of Tests 5 and 6. Not only will the results of this test provide execution times on a useful and probably frequent operation, it will also tell whether a multiply can be chained with whatever method the machine designer has chosen to implement the summation function.

7. Scalar*Vector + Vector

```
        Do 700 i = 1, N
700     X(i) = X(i) + A*Y(i)
```

Although Test 7 may seem artificial, it is included because it is frequently used in matrix inversions and multiplications. It is sometimes called a "SAXPY" operation (or "DAXPY" in the double precision case). Also, the LINPACK benchmark appears to spend most of its time executing code identical to Test 7; comparison of Test 7 with the other tests above can thus be used to gauge the validity of the LINPACK numbers as a guide to processor performance.

8. Longer Expression (Add, Multiply, Add, Multiply)

```
        Do 800 i = 1, N
800     X(i) = A(i)*(B(i) + C(i)*(D(i) +E(i)))
```

With several different vector architectures available (including register to register, memory to register, memory to memory, and perhaps others), a processor's efficiency on a simple vector instruction may not directly indicate its performance on a long series of vector instructions. The purpose of Test 8 is to investigate a processor's efficiency on a more complicated expression than those above.

Some machines have single op-codes to perform most (or all) of tests 1 through 7; but none (to our knowledge) has a single instruction to implement the above expression. Inspection of the resulting instruction rates can determine whether a processor's chaining ability extends to more complex situations.

4. SCALAR LOOP UNROLLING

The scalar FORTRAN loops presented above are by their nature somewhat inefficient, since they have considerable loop overhead compared with the very simple (and fast) constructs that they contain. Tests 1 through 3 may be particularly inefficient. The technique of Loop Unrolling has long been used (primarily on scalar machines) to reduce the loop overhead in such situations, especially in the scientific programming community. For example, the vector multiply operation (Test 2 above) can also be written as:

```
        Do 200 i = 1, N, 2
        X(i) = Y(i)* Z(i)
200     X(i+l) = Y(i+l) * Z(i+l)
```

Where N must now be a multiple of two. The loop performs two multiplies at a time, thus cutting the overhead per operation in half (and in addition producing more efficient operation on pipelined machines).

"Loop unrolling" can obviously be implemented with more operations per loop--4, 8, and even 16 appear in some scientific applications. The vectorized code, however, remains exactly the same, since the functions of the loops are identical. Thus comparing vector performance with with the performance of unrolled scalar loops will show the scalar processor in a more favorable light. One would probably expect that Test 1 above would benefit most from loop unrolling, and that Test 8 would benefit least.

Is this favorable view of scalar performance realistic? In most cases, yes: even when the applications being considered for vectorization have not been coded with unrolled loops, using unrolled loop times in this benchmark is nevertheless valid--because very few application programs have only one operation in a loop. In real-world programs, which have complex loops, there is naturally less loop overhead per useful operation--and, therefore, a benchmark having unrolled loops will more accurately reflect reality.

For those reasons, the benchmark reported herein has been developed not only with pure scalar loops like those shown in Tests 1 through 8, but also with each test unrolled with 2, 5, and 10 iterations per pass. (Those unrolling depths were chosen rather than the more traditional powers of two because the vector lengths in the benchmark are 1, 2, and 5 times a power of 10). Interestingly, some machines show more scalar improvement with unrolled loops than do other machines (Ref. 1); therefore, the scalar tests provide useful information even on machines without vector processors.

5. CONCLUSION

With many performance problems unique to vector processing, it is imperative that CPE professionals in vector installations become aware of the implications of vector processing. The benchmark described herein appears as a necessary and very informative first step in that process.

CMG attendees are invited to run this benchmark free of charge, provided only that they return data to the author so that more information can be gathered on vector processors. It is actually not necessary to run the benchmark on a vector processor; s mentioned above, interesting data are produced even for scalar processors. A request form for a copy of the vector benchmark is attached.

6. REFERENCES

1. D. S. Lindsay, "Performance Architecture Comparison of Vector Processors on Large Scientific and Commercial Machines", CMG-87 Conference Proceedings.

2. T. W. Keller & D. S. Lindsay, "Development of a Performance Evaluation Methodology for the CRAY-I", 1977 SIGMETRICS/CMG Proceedings, pp. 169 - 176.

3. K. E. Jordan, "Performance Comparison of Large-Scale Scientific Computers", IEEE Computer, pp. 10 - 23, Mar 87.

4. J. J. Dongarra, "Performance of Various Computers Using Standard Linear Equation Software in a FORTRAN Environment", Argonne National Laboratory Technical Memorandum No. 23, Mar 87.

5. F. H. McMahon, "The Livermore FORTRAN Kernels: A Computer Test of the Numerical Performance Range", Lawrence Livermore National Laboratory Report No. UCRL-53745, Dec 86.

6. D. S. Lindsay & T. E. Bell, "Directed Benchmarks for CPU Architecture Evaluation", CMG-86 Conference Proceedings, pp. 379 - 385.

BENCHMARK REQUEST FORM

Name _____

Company _____

Address _____

City/State/Zip _____

Telephone (___)_____

I was an attendee at CMG-87. I enclose a 51/4" floppy diskette for a copy of (1) the FORTRAN source for the Vector Architecture Benchmark, (2) the input file, and (3) instructions.

In return for these I agree to provide within 30 days:

1. Execution times for at least one computer. NOTE: It is not necessary to run this benchmark on a vector processor;

2. Information about test conditions including (at least):
   a. Whether stand-alone or normal production run;
   b. Job priority;
   c. Machine model and characteristics (e.g.,memory);
   d. Compiler and version;
   e. Whether vector or scalar. For scalar, unrolling level;
3. Listing of timer routine used if not already in benchmark source;
4. Permission to report these data.

I understand that future versions of this Vector Architecture Benchmark may include the names of individuals who fail to meet this agreement, and I would welcome such public exposure should I fail to provide the agreed materials within 30 days.

_____
Signature

_____
Date

Mail to:

Dr. David S. Lindsay
National Advanced Systems
Mail Stop 32/75
PO Box 54996
Santa Clara, CA 95054
(408) 970-1030

3480 PERFORMANCE AND CAPACITY PLANNING

Ted C. Keller
Yellow Freight System, Inc.

ABSTRACT

At Yellow Freight our on-line system runs about 23 hours each day. Once a day, we shut our system down, back up our on-line files, and attempt to get back up as quickly as possible. Because of the nature of our system, there is no good time to shut the system down, and we are under constant pressure to keep daily shutdown to an hour or less.

About two years ago, we upgraded from old 3420-type tape drives to the then relatively new 3480 buffered tape units. We expected the anticipated improvements in both performance and reliability to reduce shutdown by 10 to 15 minutes each day.

We were not disappointed by the performance improvements we experienced. Having been quite constrained by our old tape system, we found that not only could jobs run more quickly, but there were also significantly fewer tape errors. After installing the 3480's and then rearranging our schedule to take advantage of the improved performance, we were able to decrease our shutdown period from over 70 minutes to under 50 minutes. Overall, our conversion experience was similar to that described in the Washington Systems Center technical bulletin IBM 3480 Tape Subsystem: Performance Considerations. In the context of our needs, upgrading to 3480's was a positive experience.

As with anything else, the size and scope of our on-line system has continued to expand, and with it, the amount of data to be backed up each day. The duration of our shutdown period began to lengthen, and we suspected that the capacity of the 3480 subsystem might be part of the problem. To help us determine how well our 3480 subsystem was performing, we decided to run a series of benchmarks to provide information about the following important performance issues:

How much work a 3480 subsystem could accomplish;

The impact on through-put of having various numbers of drives active;

The statistical indicators available to measure performance and what they really mean.

In this article I will first briefly describe the functional characteristics of the 3480 tape subsystem, especially those related to performance. Then I will describe how my benchmark was conducted and the results I observed. Finally, I will summarize 3480 performance and performance indicators.

CHARACTERISTICS OF THE 3480 TAPE SUBSYSTEM

The 3480 tape system consists of A22 control units and B22 tape units. The A22 control units are microprocessor driven and each contains 512 kilobytes (Kb) of memory for use in buffering operations. (Newer units come with 1024 Kb, and EC changes are being made available to upgrade older models). Two A22 control units may be connected in a dual configuration mode, and when this is done, each controller can access any tape unit connected to either controller and the buffer storage in each controller.

The microprocessor in the controller manages buffer storage. Memory in the buffer is divided into 8 **buffer segments** of 64 Kb (on older units) or 128 Kb (on newer or upgraded units). One or two buffer segments will be assigned to each active drive. The microprocessor will dynamically assign and reassign buffer segments to different drives depending on activity. When a drive is assigned two buffer segments, they will always be in the same controller and occupy contiguous storage. If most of the activity for a drive is received by one controller, the controllers will attempt to assign the drive's buffer in that controller.

In the normal buffered mode of operation, the channel accesses data only from the controller's buffers. Buffer storage in each controller can be accessed by only one channel at a time. While one controller is transferring data to a channel from buffer storage, no other channel can access buffer storage in that controller.

In a dual configuration, buffer storage will normally be assigned in the controller receiving the first I/O request. This is done to minimize the overhead incurred when one controller needs to access buffer storage in the other controller.

When data is written to the 3480, the controller returns device end and channel end as soon as the data is transferred to the buffer. No further indication is given to the host when the data is finally written to the tape media. (The exception to this is when the unit is operating in Tape-Write-Immediate mode, which is beyond the scope of this paper.) Bufferring allows the application to proceed without waiting for the actual transfer of data to the tape. When data is being read, the unit will read several blocks ahead, pre-staging data in the buffer in anticipation of subsequent read requests from the application.

In the 3480 subsystem, the controller will disconnect from the channel whenever data cannot be transferred to or from the buffer because the assigned buffer is full (write operations) or empty (read operations). The controller will also disconnect from the channel if the buffer being accessed is not available for any of a number of other reasons, such as the buffer storage being accessed by another channel or the buffer being reorganized to adjust the number of buffer segments assigned. The performance measure **disconnect time**, as reported by RMF, is that portion of I/O service time that the channel is disconnected from the controller because the buffer could not be accessed.

On a write operation, several blocks of data in the buffer may be transferred to the tape in a continuous operation. Similarly, on a read operation, multiple blocks can be read into the buffer in a continuous operation. This type of processing, called **streaming**, is considerably more efficient than starting and stopping the tape for each operation. Whenever possible, the controller will attempt to allow devices to process in streaming mode.

The controller, however, will not let one tape drive dominate the subsystem. Since only one drive can access the buffer storage in a controller at a time, the controller may need to stop streaming activity on one drive to allow other drives to access buffer storage. The decision to stop a drive will be made based upon how much data remains to be transferred and the overall amount of activity in the subsystem. When the controller does stop a drive, it times the starting of another drive so that it will be ready to begin transferring data just after the drive being stopped completes its transfer.

Mechanically, 3480 tape drives are slower than the old 3420 type drives. Additionally, since the IBG is so small, the tape will actually move beyond the beginning of the next block whenever the drive stops. To compensate for this, the drive will automatically back up and reposition itself before the IBG each time it stops. This automatic repositioning is called back-hitch. Since the 3480 has relatively slow mechanical start and stop times, it is not as efficient when it must start and stop frequently.

HOW THE BENCHMARK WAS ACCOMPLISHED

We wrote a program which looped writing a specified number of blocks to tape. A crude synchronizing routine was included to allow all tape drives in each test to begin writing at about the same time. Enough data was written to each tape so that it was just over 90 percent full. A blocksize of 16 Kb was used, since that was typical of blocksizes used during our shutdown period. Each tape was written by a separate job, and each job was made non-swappable and given the same dispatching priority. Tests were run using from one to 16 tapes.

The benchmark jobs were run on a pair of 3090 model 200 processors using MVS/XA. One processor was dedicated to the test; the other was processing a minimal CICS workload. Two A22 controllers in a dual configuration were used, each with four attached B22 tape units. Selected drives were varied on-line in different tests to allow selection of controllers. The initial benchmark was run with 512 Kb of buffer storage in each controller. A later benchmark was run after the controllers were upgraded to 1024 Kb, and a summary of the differences is provided in Appendix A.
RMF was used to record various statistical data such as channel and device activity. The reporting interval was set at one minute. When the results were interpreted, only those RMF samples in which all tapes were active for at least 58 seconds of the minute interval were used. (For example, if a particular benchmark run involved using 8 tapes, the only RMF intervals used for most analyses were those minute-long periods in which all eight tapes were active for at least 58 seconds.) This was done to eliminate partial data which could distort statistics.

THE RESULTS OF THE BENCHMARK

The first group of tests to be analyzed used only one tape drive at a time. In these tests, run times varied from 78 - 88 seconds, with a mean of 82 seconds and a median of 80 seconds. (For our purposes, run time includes only the time after file opens and job synchronization that jobs were actually attempting to write to tape.) Separate tests were run on each of the 3090 processors and the results were comparable.

RMF data showed service times for some typical runs (those with run times about 80 seconds) to be 79 - 80 ms. Thus it took about 1000 start subchannel commands to transfer the 10500 blocks, or about 10.5 blocks per start subchannel. Our effective service time was about 7.5 ms per 16 Kb block. During these tests, disconnect time averaged 8.3 - 8.6 ms per I/O or about 0.8 ms per block. This represented about 10.5 percent of service.

The disconnect time probably occurred because the channel was somehow able to transfer data slightly more quickly than it could be transferred to the device. Overall, we were able to transfer about 2.1 - 2.2 megabytes per second out of a theoretical maximum of 3.0. It is probably safe to assume that the device streamed most of the time. In almost every way service was good and, for those used to 3420-class drives, surprisingly fast.

The second group of tests involved the use of two tape drives. Tests were run:

a. on one processor with both tapes on the same controller;
b. on one processor with the tapes on separate controllers;
c. on two processors with the tapes on separate controllers.

The results of these tests were somewhat surprising. To begin with, runs using two tape drives on the same controller followed an unusual pattern. Each run took either 141 - 144 seconds or 114 - 118 seconds, but no run took between 119 and 140 seconds. Similarly, when the tapes drives were on separate controllers, run times clustered in two ranges: from 107 - 113 seconds or from 127 - 136 seconds, with no runs completing between 114 and 126 seconds. When the tape drives on separate controllers were processed from different processors, all runs completed in 94 - 99 seconds.

Exploring the probable reasons for the variations in run times can help provide some interesting insights into the nature of the 3480 subsystem and how it performs.

In the third set of runs using two processors and separate controllers, there should have been very little interference between the drives. Since each of the controllers was defined with a different preferred path, activity should have been funnelled through one controller on each system, and the buffers should have been allocated in separate controllers. There should have been no contention for the channels (different channels on different systems) and no contention for buffer storage. Overall, there should have been very little internal contention and neither tape's performance should have been materially impacted by the other. In theory, we should have expected run times to be just a little longer than in earlier runs when only one tape was active.

Run time did increase by an average of about 12 seconds (or 17 percent) over tests with a single active drive. Average service time was 93 ms per start subchannel command with an effective service time of about 9.0 ms per block. Of this, approximately 2.0 ms (22 percent of service) was disconnect time. If about .8 ms of disconnect time per block could be associated with delays similar to those experienced with a single active drive, the additional 1.2 ms per block of disconnect time would seem to be due to additional controller overhead. This was probably caused by the additional processing and coordination required to keep multiple drives active.

In the first set of the tests with both drives on the same controller, MVS/XA would have attempted to use the same path for both drives (they were defined with the preferred path option to MVS/XA). The path through the second controller would have been used only when the preferred path was busy. This would have caused the buffers for both drives to have been allocated in the same controller most of the time. Since buffer storage in a controller can only be accessed by one channel and by one device at a time, the drives would had to have shared access to buffer storage. Whenever one drive was writing, the other could not; and whenever data was being transferred from a channel for one drive, the other channel was similarly locked out.

It is probably safe to conclude that in those sets of test runs completing in 141 - 144 seconds, the buffers were allocated in the same controller. RMF statistics for these runs show device service time of about 143 ms. Again, it took about 1000 start subchannel commands per tape, with about 10.5 blocks written per start subchannel. The effective service time was about 13.6 ms per block of which 6.6 ms (48 percent of service time) was spent disconnected. About 2.0 ms of the disconnect time was probably caused by overhead similar to that described above. Most of the remainder of the disconnect time (about 4.6 ms per block) was probably caused by contention for buffer storage.

But not all runs using two tape drives on the same controller experienced the same elapsed times; runs in some tests completed in only 114 - 118 seconds. The likely reason for this is that, in these runs, the buffers would have been allocated in separate controllers. If the preferred path was busy doing work for the first tape when the initial write command was issued for the second tape drive, then the alternate path through the second controller would have been used, and the buffers would have been allocated there. With buffers allocated in different controllers, the contention for buffer storage was eliminated. The few times this happened, elapsed times decreased by about 26 - 27 seconds.

In test runs using two tapes on separate controllers, run times were slightly better than when both drives were on the same controller. Run times again clustered in two ranges, 107 - 113 seconds and 127 - 136 seconds. In these tests, since each controller would have been assigned a different preferred path, buffers tended to be allocated in different controllers. Only when one of the preferred paths was busy at the time of the first I/O request would the two buffers have been allocated in the same controller.

We can again see the impact of buffer placement in these test runs, but can also see the impact of two other factors. In these tests, when the buffers were in opposite controllers, most I/O requests were being received by the controller containing the buffer segments. Because of the reduced inter-controller overhead, these test runs completed in an average of 5 - 7 seconds less than the best run times when the two drives were connected to the same controller. Additionally, even when the buffers may have been allocated in the same controller initially, it is likely that the controllers' load-balancing algorithm moved one of the buffers to the other controller. This should happen since the predominance of each tape's work would flow through a different path, and the controllers would attempt to locate buffers where most of the activity was occuring.

Furthermore, with both tapes on the same controller, MVS/XA would have tried to schedule most I/O's for both drives over the same path and should have used one channel more heavily than the other; with tapes on separate control units, activity should have been more evenly balanced between the two channels. In many cases one half of the channel pair was accessed significantly more heavily than the other when both drives were on the same controller, and the channels tended to be more evenly utilized when drives were used from both controllers.

When four or more drives were active, the results were similar to those with two active drives except that run times and device service times became longer as more drives were active, and there were wider variations in run times for reasons explained above. On the average, disconnect time increased to about 63 percent of service time with four drives active, but this varied considerably depending upon several factors. For test runs conducted with additional active drives, service times continued to increase as did disconnect time per block written. With 12 drives active, disconnect time accounted for about 89 percent of service; and with 16 active drives, about 91 percent.

With four active tape drives, chances became good that some buffers would be allocated in each controller. It was unlikely that all four drives

would have their buffers allocated in the same controller. In most cases, each controller would have the buffers for two or three drives, regardless of which drives were used. With five or more active drives, each using two buffer segments (as long as buffer storage would be available in one of the controllers, the controllers would try to allow each active drive to use two buffer segments), some buffers had to be allocated in each of the controllers. With multiple active drives, sometimes the workload appeared to be balanced between the controllers and sometimes it appeared quite unbalanced. It was not uncommon to see one or two tapes finish as much as 100 seconds sooner than the other tapes in the same test when larger numbers of tape drives were used.

With four active drives on one processor, channels were 80 - 90 percent busy during most tests. Additionally, subsystem connect time per second (the time a channel was connected to a controller transferring data) totalled between 100 and 104 seconds per minute. The tape subsystem was being used to about 86 percent of its theoretical capacity.

Overall, subsystem throughput increased as more tapes became active but seemed to peak with six active drives. About the same amount of work could be done per unit of time with anywhere between five and eight active drives. With nine or more active drives, throughput began to fall (both total connect time per minute and channel busy measures decreased.) Less total work per minute took place with 16 active drives than with three.

With nine or more active drives, some drives would have to be assigned less than two buffer segments. Normal controller overhead also continued to increase with more drives. With large numbers of active drives it is also quite likely that buffer segments would be stolen from some drives and reassigned to others that may have appeared more active from time to time.

Five to eight active drives seemed to provide maximum levels of throughput. (In some cases, this same level of work could be accomplished with only four active tapes.) With five active drives on one processor, the sum of channel busy percentages for all active channels was normally over 180 percent. Total connect time was about 110 seconds per minute. Total channel busy and total connect time per minute remained fairly constant with five to eight active drives. But, of course, even though the amount of work accomplished per unit of time remained stable in this range, run times and block service times increased when more drives were active.

CONCLUSIONS AND SUMMARY

The 3480 tape subsystem is powerful but complex. Factors over which we have little or no control, such as the location of buffers, can have a significant impact on performance. It is difficult to obtain completely repeatable results, even in a closely controlled benchmark. Because of this, it is difficult to accurately assess performance experienced in any given situation. Instead, when analyzing the 3480 subsystem, it is important to review the results of several related or similar periods.

The best indicators of total work accomplished (throughput) were total channel busy (of all tape channels in use) and total connect time per minute. With five to eight active drives, total channel busy averaged just over 180 percent, and total connect time ranged from 106 to 110 seconds per minute. With six active drives, total connect time and channel busy normally were just a little higher than with any other number of active drives. Figure 1 shows total channel busy values for different numbers of active drives.

It would seem from total channel busy figures that throughput might have peaked at about 5.5 million characters per second (cps) (3 million cps * 2 controllers * 95%). However, effective throughput (total characters transferred to tape by all runs of a test divided by the mean run time of all jobs in that test - remember that our run times include only transfer times and exclude mount, open, and rewind times) was always considerably less. No run achieved effective throughput better than about 4.6 million cps. For runs with a single active drive, the best effective transfer rate was about 2.2 million cps; total channel busy measured during that run was 90.8 percent, which at a 3 million character transfer rate would seem to have been about 2.7 million characters per second. With two active drives, the best effective transfer rate was 3.6 million cps, while channel busy values suggested higher performance. Throughout all of these tests, effective transfer rate peaked at 2.2 - 2.3 million cps per controller.

Jobs ran faster with fewer active drives. With only one active drive, a tape could be written over 90 percent full in about 80 seconds. With two active drives run times climbed to 94 - 144 seconds, depending on whether one or two processors were used, where the buffers were allocated, and on which paths the drives were located. Run times continued to lengthen with additional active drives, and were as high as 850 - 950 seconds with sixteen active drives. Figure 2 shows run times for different numbers of active drives.

The best indicators of performance appear to be the ratios of connect time or disconnect time to device service time. Connect time represents the portion of service time that the channel was connected to the buffer doing useful work; disconnect time is the portion of service time during which the buffer was not available. Normally, unless a tape is being mounted or rewinding, connect time and disconnect time together will account for 99 percent or more of service time. In general, the higher the connect time ratio or the lower the disconnect time ratio, the better the performance the device is receiving.

Connect time ratios ranged from about 90 percent with a single active drive to about 8 percent with sixteen drives active, while disconnect ratios ranged from about 10 to 91 percent, respectively. Connect ratios of 20 to 35 percent were common during periods of high system throughput (five to eight active drives). Figure 3 shows the correlation between connect time ratios and total run times. Figure 4 shows connect time ratios for minute long intervals, and Figure 4-A shows the mean

connect time ratios for various numbers of active drives.

A number of other tests were run to briefly explore other aspects of performance. In one test, we increased the blocksize to 32 Kb; here run times improved by about 8 percent with a single active drive, and total effective throughput climbed to 2.4 million cps. In some tests we tried smaller blocksizes (2 Kb) and mixing runs using both files with large and small blocksizes. In these tests, we found that not only did it take longer to fill a volume with smaller blocks, but that in the tests using both large and small blocksizes, the smaller blocksizes hindered the writing of larger blocks. In some later runs, we tried reducing the number of host buffers (DCB=BUFNO) from 20 to 8; we found no noticeable difference with one active drive, but detected slight improvements in elapsed time with multiple active drives.

This study and benchmark concentrated on 3480 subsystem performance when writing large amounts of data to tape. All files in my tests were being pushed as hard as possible. My tests purposely excluded mount, open, and rewind times. In most environments, though, we do not usually find such levels of stress by all active drives. Most jobs have other things to do besides write data to tape and will not normally attempt to write data as intensely as our tests did. It is also probable that some number of drives will be loading or rewinding at any given time. These tests were designed to determine the processing limits of the 3480 subsystem when writing large volumes of data, and may not be completely suitable for analysis of tape workloads which include a wide range of various kinds of activity.

While I have discussed the operating characteristics of the 3480 subsystem, described a series of benchmarks, and summarized their results, I have not yet demonstrated how all this information can be useful. In Appendix B, I have included the results of a study I recently completed relating 3480 performance to specific capacity analysis.

REFERENCES

1 Grossman, G. L. and Long, R. G., Performance Evaluation of Buffer Tape Subsystems, CMG Transactions, Fall, 1986.

2 IBM 3480 Magnetic Tape Subsystem Planning and Migration Guide, IBM Manual GC35-0098-4, January 1987.

3 Middleton, G. F. and Taylor, L. B., IBM 3480 Tape Subsystem: Performance Considerations, IBM WSC Technical Bulletin GG22-9335-00, August 1985.

4 Seiderer, R. J., 3480 Installation Guide and Reference, IBM WSC Technical Bulletin GG22-9399-00, May 1985.

APPENDIX A

SUMMARY OF PERFORMANCE DIFFERENCES AFTER THE MEMORY UPGRADE

After memory was upgraded to 1024 Kb on each of our 3480 A22 units, I reran many of the benchmark runs. As might be expected, there was some improvement at all levels of processing.

With a single active tape drive, run times for eight separate runs were all between 75 to 77 seconds. This was not only a little better than all similar previous runs, but more consistent; run times had ranged from 78 to 88 seconds with a mean of 82 seconds. Effective transfer rate now peaked at 2.3 million cps versus 2.2 million cps previously. Total channel busy was about 94% during these runs, compared to a peak of 88% before the upgrade. Connect time was 95 - 96% of service, up from 90% previously. Figure 5 shows run times for different numbers of active drives.

Tests with two active drives followed patterns similar to those in previous benchmarks except that run times were shorter and the impact of buffer placement appeared to be less obvious. With two drives on a single control unit, run times were 100 - 123 seconds with a possible break in ranges around 110 seconds. With the two drives on separate controllers, individual run times ranged from 87 to 122 seconds with no discernable pattern. When the two drives were accessed from separate processors, run times fell to 78 - 107 seconds. Depending upon several factors, run times varied considerably but were consistently better than before the upgrade. Channel busy values were correspondingly higher as was effective transfer rate. With as much variety as there was in each set of run times, it is hard to give meaningful percentages of improvement.

As was the case before the upgrade, total throughput peaked with six active drives (at 4.9 million cps). Throughput was consistently good with up to eight active drives and did not decline materially until more than 12 drives were active. In fact, even with 16 active drives, total channel busy was over 170 percent and the effective transfer rate was over 4.2 million cps.

The ratio of connect or disconnect times to service time continued to be a good indicator of performance. With a single active drive, connect/service ratios averaged 95 - 96%. In periods of high throughput, ratios of 35 - 50% were common. With 16 active drives, 10 - 13% was typical. Figures 6 and 7 show the relationship of connect time ratios to run times and to the number of active drives.

The addition of memory to the 3480 controllers appears to have helped performance, especially when more than eight drives were active. While only improving run times by less than 10 percent with four or less drives active, the additional memory facilitated streaming when larger numbers of drives were active and permitted vastly improved levels of throughput at higher levels of activity. It is easy to see in Figures 2 and 5 the differences in performance after the upgrade.

APPENDIX B

A CASE STUDY IN 3480 PERFORMANCE

Once a day, we shut down our on-line system and create backups of our on-line data. The time that we are down is critical, and we attempt to minimize it as much as possible. At the time of this study, the shutdown period had grown just beyond critical limits, and we were interested in determining whether additional 3480 controllers could be of any benefit. This study was performed after memory in the controllers had been updated. Benchmark data referenced was from runs using the upgraded controllers.

The first step taken was to run Omegamon with DEXAN on several selected runs for several days. This would tell us the major components of the jobs' run times. Unless tape processing delays were a considerable percent of run time, there would be no need to look any further.

DEXAN showed us that at least 25 - 36% of the wait time for these jobs was associated with tape processing. A slightly larger percent was for DASD processing, and about 15 - 20% was caused by other delays. These percentages varied from job to job and from day to day, and at times tape processing accounted for as much as 50% of response time, but was seldom less than 25%. The limit of benefits to be expected from tuning 3480 processing would be about 25 - 36% of processing time.

The next step in my analysis was to review RMF statistics for the 3480 devices and channels. For this special analysis, we shortened RMF's reporting interval to one minute and increased sampling to 10 times per second. The reason a one minute interval was selected was that individual jobs might run for only 10 to fifteen minutes, and most drives would process several different tapes and even different files within a ten to fifteen minute period.

RMF showed that connect time was normally 20 - 30% of service for most drives in use during shutdown. While this varied considerably from minute to minute and day to day, this range of performance tended to be common during a large portion of our shutdown period, and there were many periods of five to ten minutes in which connect time ratios of active drives seldom rose above 20 - 25%.

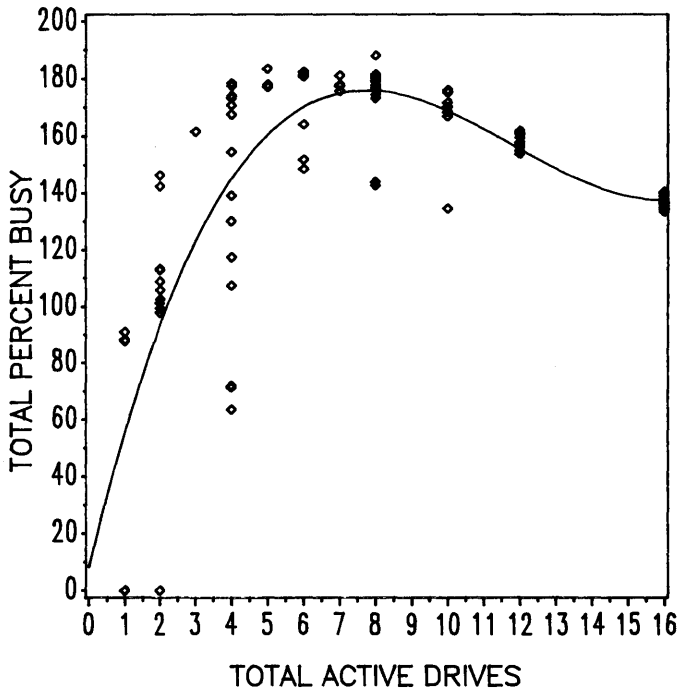
Using this information and information from Figure 7, I saw that connect-service ratios of 20 - 30% corresponded to a level of work generated by 6.5 to 10 very active drives. If this work could be fairly evenly split with another controller, we should expect the level of activity to drop approximately in half. As shown in Figure 8, this should result in an improvement in the connect-service ratio to about 38 - 50%.

Looking at Figure 9 we can see that the projected improvement in the connect time should result in an improvement in the amount of work we should be able to accomplish. Using the correlation between run times and average connect-service ratios as an approximate indicator, we should reduce tape delays by about 50%. It is somewhat difficult to estimate exactly how much improvement might be expected because of the wide range of performance experienced in controlled benchmark runs.

Since only about 25 to 36% of the total elapsed time was attributable to 3480 processing, we could potentially reduce our processing time on key jobs by 13 to 18 percent. Further examination of our shutdown period showed that only about 40 minutes were truly "tape intensive." From this it appears that shutdown could be reduced by 5 - 7 minutes if two A22 controllers were added.

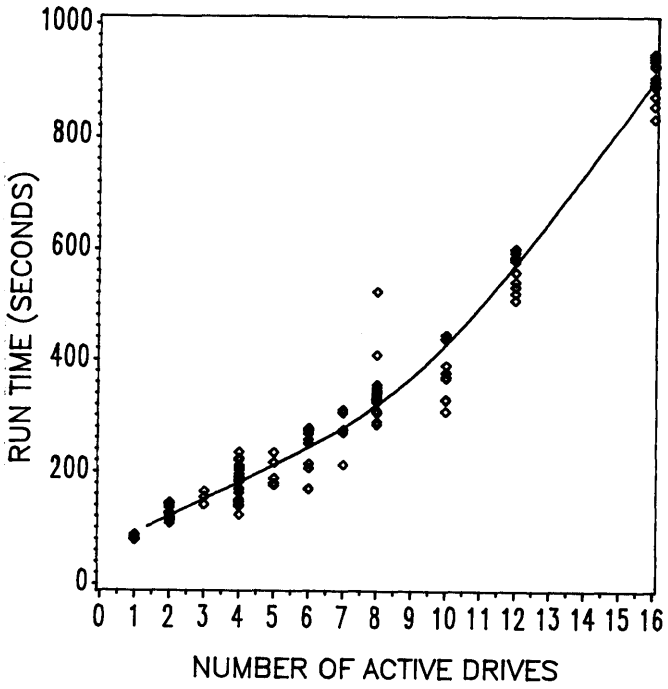
Similar calculations were performed analyzing the impact of proposed expansion to our on-line system. Additional volumes would need to be backed up each day, and those back-ups could be run either after existing jobs completed (thus directly extending our shutdown period) or at the same time existing jobs were running using the 2 - 3 tapedrives that are free during most of the shutdown period. Using a technique similar to that described above, I figured that running two or three additional tape intensive jobs could extend shutdown by 3 to 10 minutes each day if the additional controllers were not added.

TOTAL CHANNEL BUSY VS. TOTAL ACTIVE DRIVES



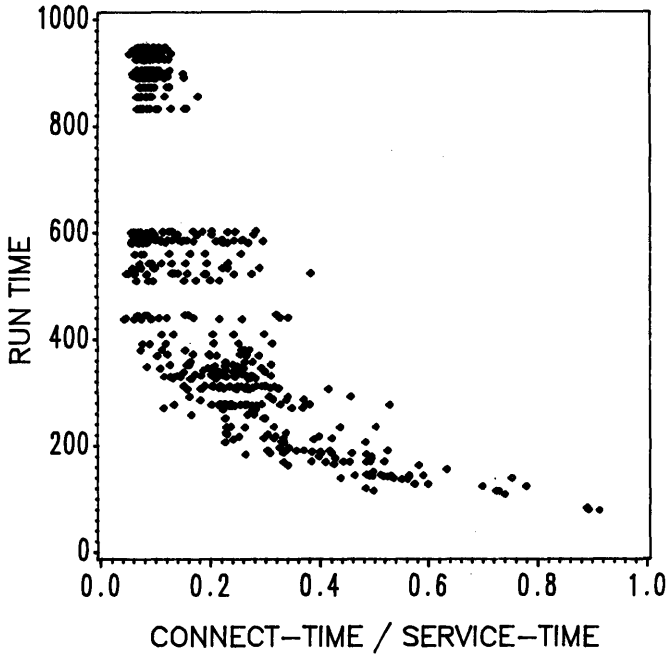
SUM OF CHANNEL BUSY VALUES.
BENCHMARK RUNS PRIOR TO MEMORY UPGRADE.
FIGURE 1

BENCHMARK RUN TIMES



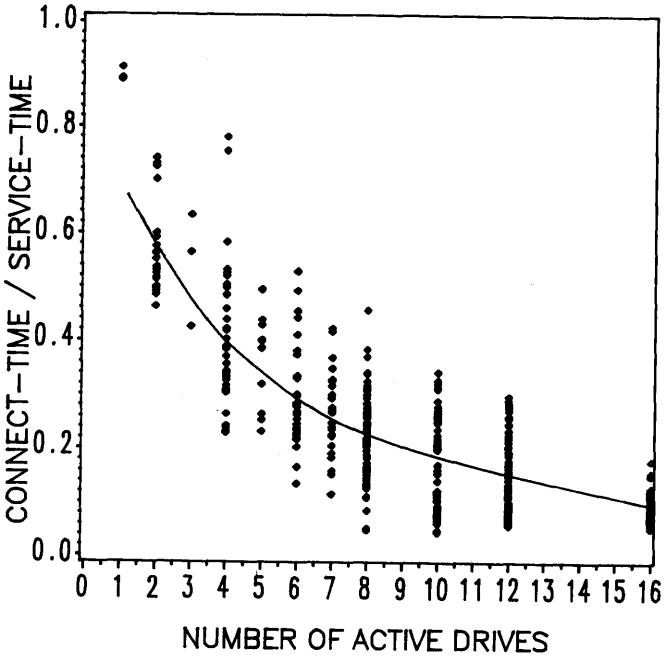
NUMBER OF SECONDS TO WRITE DATA TO TAPE.
BENCHMARK RUNS PRIOR TO MEMORY UPGRADE.
FIGURE 2

CONNECT TIME VS RUN TIMES



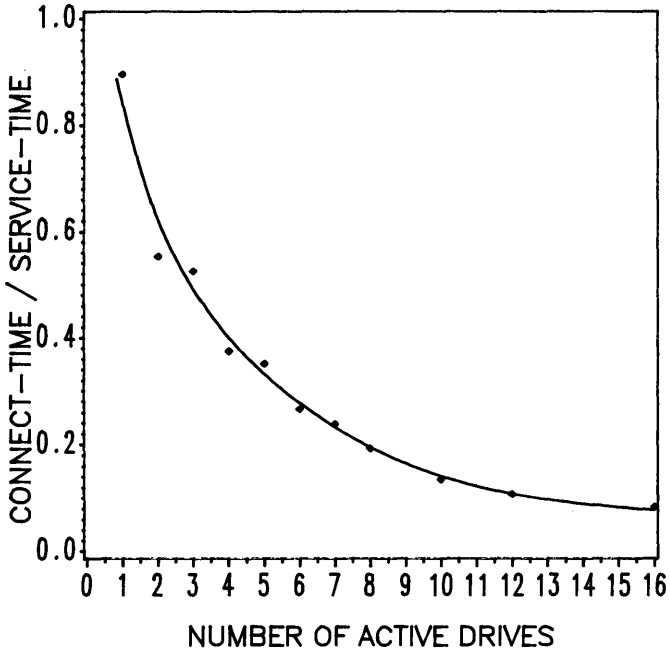
THE CORRELATION BETWEEN RUN TIME AND
THE CONNECT–TIME / SERVICE–TIME RATIO.
BENCHMARK RUNS PRIOR TO MEMORY UPGRADE.
FIGURE 3

CONNECT TIME RATIO VS. ACTIVE DRIVES



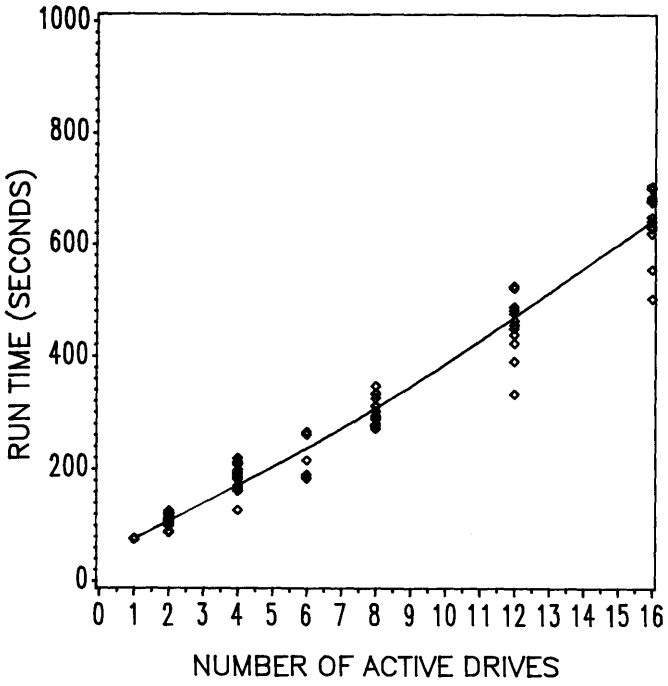
CONNECT TIME RATIO VERSUS TOTAL NUMBER
OF ACTIVE DRIVES.
BENCHMARK RUNS PRIOR TO MEMORY UPGRADE.
FIGURE 4

CONNECT TIME RATIO VS. ACTIVE DRIVES



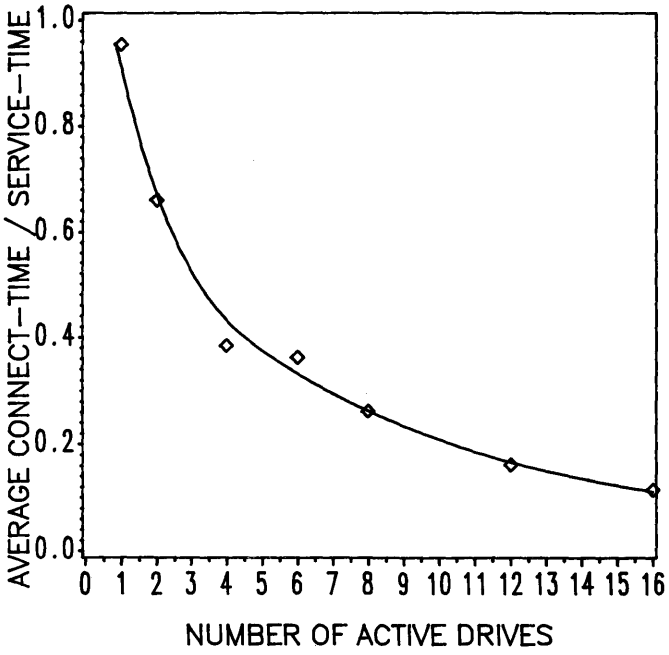
AVERAGE CONNECT TIME RATIO VERSUS TOTAL
NUMBER OF ACTIVE DRIVES.
BENCHMARK RUNS PRIOR TO MEMORY UPGRADE.
FIGURE 4–A

BENCHMARK RUN TIMES



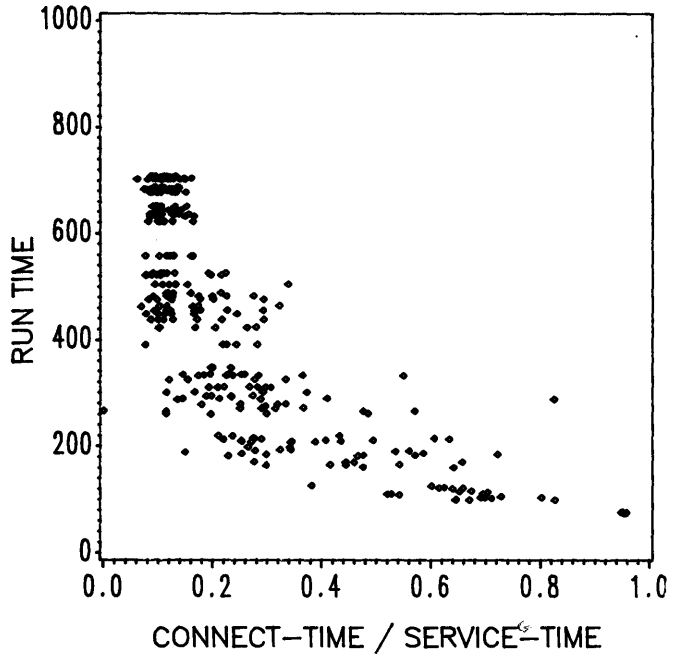
NUMBER OF SECONDS TO WRITE DATA TO TAPE.
BENCHMARK RUNS AFTER MEMORY UPGRADE.
FIGURE 5

AVERAGE CONNECT TIME RATIO VS. ACTIVE DRIVES



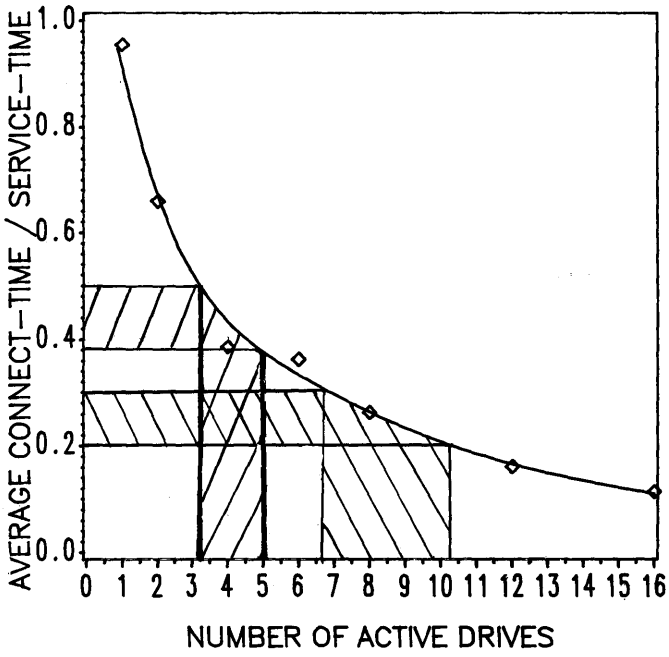
AVERAGE CONNECT TIME RATIO
VERSUS TOTAL NUMBER OF ACTIVE DRIVES.
BENCHMARK RUNS AFTER MEMORY UPGRADE.
FIGURE 7

CONNECT TIME VS RUN TIMES



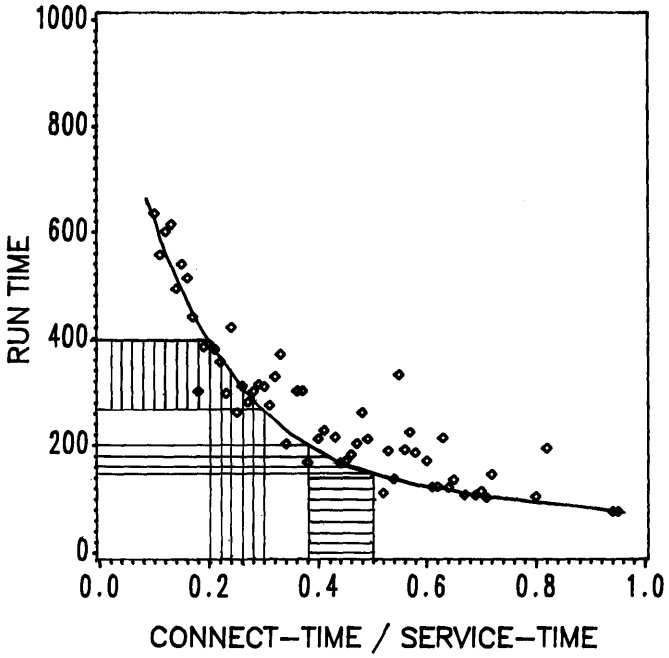
THE CORRELATION BETWEEN RUN TIME AND
THE CONNECT-TIME / SERVICE-TIME RATIO.
BENCHMARK RUNS AFTER MEMORY UPGRADE.
FIGURE 6

ESTIMATION OF CHANGE IN CONNECT-TIME RATIO



ESTIMATE OF CHANGES IN CONNECT-TIME RATIO
WHEN EQUIVALENT NUMBER OF DRIVES DECREASES.
BENCHMARK DATA AFTER MEMORY UPGRADE.
FIGURE 8

ESTIMATE IN CHANGE IN 3480 PERFORMANCE



ESTIMATE IN RELATVE CHANGES IN 3480 PROCESSING TIMES BASED ON CHANGES IN CONNECT–RESPONSE TIME RATIOS.

FIGURE 9

Transaction Resource Demand

James T. Cheeseman
3M

ABSTRACT

Transaction chargeback is a matter of increasing concern. Most current charging schemes fall within two general categories:

- Fixed Transaction Cost
- Variable Transaction Cost Based On Resources Used

Both of these methods have their own set of problems, but neither of the methods provides a real measure of Resource Demand - how difficult will it be to provide resources for this set of transactions such that the required response time performance objective is met?

This paper presents a method which provides a quantitative measure of Resource Demand. It includes known parameters which can directly gauge the level of resources needed to service transactions in an interactive transaction oriented system.

Using the measure described in this paper, transactions can be assigned to groups which have similar Resource Service Demands. Charges can then be assigned to relate transaction cost to the degree of system resource consumption.

If chargeback is not the central issue, this measure can be used to rank transactions into groups which warrant special treatment to minimize particular performance problems.

BACKGROUND

With the initial installation of the Information Management System (IMS[1]) at 3M, a recharge method was implemented to apply a fixed charge per IMS transaction. As the IMS workload continued to increase, and the disparity between transactions increased, it became apparent that the method of charging must change to more accurately reflect the resource demand of each transaction. Resource demand is a good measure because it quantifies the actual demand on system resources necessary to meet established response time performance objectives for the transactions.

Discussed in this paper is a transaction charging technique that determines transaction charges based on the physical resource demand of the transactions. Although this paper is directed towards IMS, this method would apply to any transaction oriented system which has established response time performance objectives.

Charging Algorithms

Most charging algorithms can be reduced to two general methods to accomplish transaction charging:

1. Fixed Charge

   This is probably the simplest method of charging - each scheduling of a transaction costs a fixed amount. This is a good charging method if all transactions are created equal. However, since a transaction costs the same amount regardless of how much work is done, there is a temptation to make each individual transaction do more and more work. This causes a large disparity in transaction resource demand, and does not result in an equitable charging method since the longer running transactions are subsidized by the shorter running transactions. This also leads to scheduling difficulties since a greater number of IMS Message Processing Regions (i.e., Resources) are needed to guarantee meeting response time performance objectives for the longer running transactions.

2. Variable Charge

   This is probably the most equitable method since each scheduling of a transaction is charged based on the number of actual service units (or some other measure) consumed. The problem with this method is that each time a transaction is scheduled, the charge is different based on the actual amount of work done. This method makes it extremely difficult to forecast application system costs. There are also implications involving the cost per transaction computation - as the amount of work done by each transaction varies from month to month, the cost per transaction will also vary based on the budget amount to be recovered.

Transaction Resource Demand

A method has been developed which not only offers a compromise between these two methods, but is superior because it gauges the resource demand of transactions. In other words, how much demand is placed on system resources to service the transactions and guarantee that established Response Time Performance Objectives will be met?

The formula, derived in Appendix A, is:

$$F = \frac{A * R * E}{R - E}$$

Where F = Resource Demand Factor
A = Peak Hour Arrival Rate
R = Response Time Performance Objective
E = Transaction Residency Time

The Resource Demand Factor represents

> "The number (or fraction) of message processing regions required to sustain the given transaction arrival rate and meet the required Response Time Performance Objective"

and is a true measure of transaction resource demand. Appendix E contains an analytic model verification of this formula.

The three parameters in this formula are known values for existing transaction workloads: the Peak Hour Arrival Rate and the Transaction Residency Time can be obtained from standard monitoring software products; the Response Time Performance Objective is obtained from customer service agreements.

The three parameters of this formula are interpreted as follows:

1. Transaction Residency Time, E

   Residency time is the amount of time a transaction is resident in a message processing region.

   Residency time is a good measure since it inherently includes database I/O (the more I/Os done, the longer the residency time), database contention (waiting for I/O), and processor time. Since residency time is measured from the point when a transaction starts execution, scheduling delays are not a part of this time. A Program Reuse option, or the use of a program preload software product such as Quick-Fetch[2] or Virtual Fetch[3], has the effect of reducing average residency time since the average application program load time is reduced or nonexistent.

   It should be noted that residency time is adversely affected by the traversal of a disorganized database - extra processing time is required by the system to locate records in overflow areas. Since an increase in residency time causes an increase in cost, the tendency will be to schedule occasional database reorganizations.

2. Peak Hour Arrival Rate, A

   The Peak Hour Arrival Rate is the maximum arrival rate expected or experienced for this transaction over all hours.

   Peak Hour Arrival Rate was chosen since it provides a measure of the wors case for which resources must be provided. For example, if the Peak Hour Arrival Rate is 1 transaction per second, the resources needed to process a given set of transactions is considerably different than if the Peak Hour Arrival Rate is 10 transactions per second. The Resource Demand Formula shows a factor of 10!

3. Requested Response Time, R

   The Requested Response Time is the response time expected by the user according to an established Response Time Performance Objective.

   The Requested Response Time is necessary since it is the driving force that determines how fast a transaction must be serviced and, consequently, the amount of resources which must be provided. Given any residency time, obviously, more resources are needed to provide service for, say, 5 second response time than would be needed to provide service for 10 second response time.

   Since most response time performance objectives are stated in terms of a statistical confidence level statement such as:

   > "95 Percent of the transactions will complete in less than R seconds",

   an arbitrary choice of the confidence level to use must be made. The choice is arbitrary since the total amount recovered should approximate the cost of providing the service. Appendix D contains a discussion and derivation of a method that can be used to convert from a chosen confidence level to a different confidence level.

This formula has some rather interesting properties and implications. Since the denominator is a difference which can approach zero, the formula is of exponential form. This is seen frequently in queueing systems. The exponential form is expected since as the transaction residency time (E) approaches the response time performance objective (R), the number of message processing regions required (F) must necessarily be large.

An example will demonstrate the nonlinear behavior of this formula:

Given a peak hour arrival rate of 1 transaction per second and a residency time of 0.75 seconds, 3 message processing regions are required to meet a performance objective of 1 second. Whereas, if the residency time is 0.5 seconds, only 1 message processing region is necessary!

This demonstrates that, for these conditions, the first transaction has a residency time 50% greater than the second transaction, but it requires 3 times the resources to be processed!

However, if the Requested Response Time of the first transaction is increased to 3 seconds, then only 1 message processing region would be required.

Appendix B provides a graphical example of the comparison between two transactions that have different resource demand factors.

Boundary Conditions

Since an exponential equation has some unique properties, it is helpful to examine the boundary conditions.

In the case where the transaction residency time is equal to the response time performance objective, the resource demand factor is infinity. This means an open message processing region must be available whenever a transaction arrives in the system - otherwise a delay will occur. Thus, the number of message processing regions established must be equal to the expected maximum number of transactions that could be in the system concurrently. To have less than this number of message processing regions means that a delay will inevitably occur, causing the response time performance objective to not be met. The number of message processing regions could be slightly less if the response time performance objective was stated in terms of a confidence level for the response time performance objective.

In the case of the residency time exceeding the requested response time, the Resource Demand Formula gives a negative value. This indicates that it is impossible to provide a response time that is less than the amount of time a transaction is resident.

Transaction Resource Usage Ranges

Since it probably would not be practical to have a different charge for each unique transaction, a system of categorizing the transactions into resource usage ranges provides a more reasonable alternative. It is relatively easy to obtain average residency times and peak hour arrival rates of existing transactions over a period of time. Using these values, along with established Response Time Performance Objectives, the range of values for the Resource Demand Factor can be computed. Depending on the number of price categories desired, the range of Resource Demand Factors can be divided into Resource Usage Ranges. Given the total amount of money that must be recovered over a given period of time, along with the projected transaction volumes, a price for each category can be assigned. An example of price categories is shown in Appendix C.

It is desirable to have more than a few categories so that the price differential between categories is not excessive. A small number of price categories could have a significant effect on transactions which fall near the price category borderlines.

In the case of new transactions, the residency time can be estimated from data gathered during transaction testing or, perhaps, from a transaction estimation tool. This will allow assigning the transaction into a temporary category while long term averages are obtained.

It is suggested that long term tracking of peak hour arrival rates and residency times be done for each transaction. This will allow moving transactions between categories if the resource demand changes significantly.

Benefits Of Resource Demand Charging

Since shorter residency time transactions, for a specified response time performance objective, correspond to a lower charging rate, a tendency will be to create small, single purpose transactions to take advantage of the lower cost. From a performance standpoint, this is exactly what is desired since many short transactions place less of a demand on system resources and reduce the scheduling problems inherent in large resource demand transactions. The net effect of this is to consistently meet established performance objectives while postponing costly resource upgrades.

In a recent journal[4], it was stated that in published lists of reasons for chargeback, the item "Promote Optimal Use Of Resources" is never included. Transaction charging by resource demand does this very thing!

An example will demonstrate how Optimal Use Of Resources is promoted by the nonlinear behavior of the Resource Demand Formula. The costs used in this example are from Appendix C and the transaction types are those stated in Appendix B.

Suppose there were 100 Type B transactions which have an associated cost of $0.025/transaction. If these were converted into Type A transactions, to do the same amount of work, 250 transactions would be required. Since the cost for A type transactions is $0.005, the total cost for 250 would be $1.25 - the total cost for the B type transactions is $2.50. Consequently, the cost is 50% for doing an equivalent amount of work.

Conclusion

A method for IMS Transaction charging with a firm base in operational analysis has been developed. This method provides a transaction charging method based on a measure of resource demand which is a function of peak hour arrival rate, IMS message region residency time, and the response time performance objective. The method closely approximates the results obtained in queueing systems.

Acknowledgement

This paper is a result of discussions with T. M. Kabrick of 3M on his idea to develop a method for charging transactions based on the demand placed on system resources to meet established response time performance objectives.

I would like to thank the following people from 3M for their helpful comments and criticisms: T. W. Akemann, J. E. Knutson, M. D. Korich, E. C. Robinson, and J. M. Strom.

A special thanks to Mary Shafer for her effort in transcribing this paper to "CMG Conference Format".

APPENDIX A.

RESOURCE DEMAND FORMULA DERIVATION

This appendix provides the derivation of the Resource Demand Formula using established laws of operational analysis[5].

The response time a typical transaction will experience is:

$$R = N*E + E \qquad (1)$$

Where R = Transaction Response Time
N = The Average Number Of Transactions In The Queue
E = Transaction Residency Time

In essence, this states that the response time a particular transaction will experience will be the residency time for all transactions in the queue ahead of it plus its own residency time.

Little's Law from Queueing Theory states that the number of transactions in the queue is the product of the response time and the throughput.

$$N = R * X \qquad (2)$$

Where N = The Average Number Of Transactions In The Queue
R = Transaction Response Time
X = Throughput (Arrival Rate)

Solving equation 1 for N and equating to N in Little's Law (Equation 2) produces:

$$\frac{R - E}{E} = R * X \qquad (3)$$

Solving for X gives:

$$X = \frac{R - E}{R * E} \qquad (4)$$

If R is interpreted as the requested Response Performance Objective, then X is the maximum throughput that can be processed in a message processing region. Given a peak hour arrival rate A that must be processed, then the required degree of multiprocessing (number of message processing regions) or fraction thereof needed to process the transactions is:

$$F = \frac{A}{X} = \frac{A * R * E}{R - E}$$

APPENDIX B.

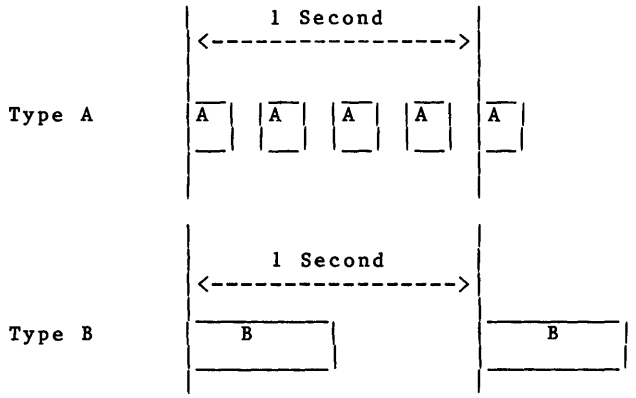
GRAPHICAL EXAMPLE OF RESOURCE DEMAND FACTOR

This appendix contains an example of the effect of transactions with different Resource Demand Factor values.

In this example, two types of transactions were chosen:

Type A	Type B
A = 1.0	A = 1.0
R = 1.0	R = 1.0
E = 0.20	E = 0.50
F = 0.25	F = 1.00

To meet the given response time performance objective of 1 second response time, the Type A transactions require one fourth of a Message Processing Region - or stated another way, four transactions of this type can be processed in one Message Processing Region in one second. The Type B transactions, however, require the full services of one Message Processing Region to meet the given 1 second response time performance objective. This criteria is represented graphically below:



Note that the Type B transactions have 2-1/2 times the residency time as the Type A transactions, but the resource demand to provide the same response time performance objective is a factor of four.

APPENDIX C.

TRANSACTION RESOURCE DEMAND COSTS

This appendix provides an example of transaction cost for the range of resource demand values.

Resource Demand	Transaction Cost
0.00-0.24	$0.005
0.24-0.41	0.015
0.41-1.00	0.025
1.00-1.50	0.050
1.50-2.00	0.070
2.00-2.50	0.090
2.50---->	0.140

APPENDIX D.

CONFIDENCE LEVEL CONVERSIONS

This appendix provides a derivation to quantify a change in confidence level for a response time performance objective.

Assume that the confidence level of the Response Time Performance Objective R in the Resource Demand Formula was chosen to be 95%. That is, 95% of the responses will be less than R seconds. If a request is made to reduce the confidence level to 90% for a given set of transactions, this could be considered equivalent to allowing the residency time of these transactions to increase by some specified amount. If this amount is quantified and used to inflate the value of R in the resource demand formula, then a relationship can be developed to convert between confidence levels.

Since the region of interest is only on values to one side of the mean residency time, the statistical confidence coefficients used are for one-sided tests. The parameter S represents the standard deviation of the residency time.

95% of the responses will be less than:

$$R = E + 1.65S$$

and 90% of the responses will be less than:

$$R = E + 1.28S$$

If we consider that the residency time is expanded, the two formulas can be combined as follows:

$$R = E + 1.65S \quad (1)$$
$$R = E' + 1.28S \quad (2)$$

$$E' - E = 1.65S - 1.28S = 0.37S \quad (3)$$

This relation quantifies the difference in residency time between the two confidence levels. This also is the same amount to inflate R to effectively moderate the response time constraint.

Using this, then, to expand R yields

$$R90 = R95 + (E' - E) = R95 + 0.37S \quad (4)$$

But, from equation (1),

$$S = \frac{R95 - E}{1.65} \quad (5)$$

Substituting this into equation (4) yields,

$$R90 = R95 + 0.37 * \frac{R95 - E}{1.65} \quad (6)$$

Rearrangement gives the result

$$R90 = \frac{2.02*R95 - 0.37E}{1.65} \quad (7)$$

which quantifies the increase for R in the Transaction Resource Demand Formula.

As an example, suppose the residency time E is 0.5 seconds and the 95% response time performance objective is 1 second. The effective 90% confidence level value for R is:

$$R90 = \frac{2.02(1) - 0.37(0.5)}{1.65} = 1.11$$

A comparison of the resource demand factor is:

$$F95 = \frac{(1)*(1)*(0.5)}{1.0 - 0.5} = 1.00$$

$$F90 = \frac{(1)*(1.11)*(0.5)}{1.11 - 0.5} = 0.91$$

This same technique would apply for conversion between any given confidence levels merely by changing the confidence coefficients in equations (1) and (2).

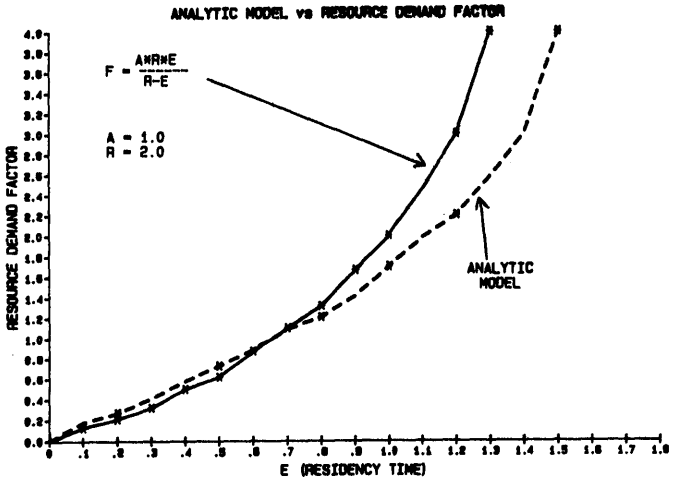
APPENDIX E.

VALIDATION OF THE RESOURCE DEMAND FORMULA

This paper would not be complete without verification that the derived formula conforms to what is expected in a computer system. Analytic models have proved their validity through the years, so a comparison of the Transaction Resource Demand Formula to an analytic model was done. The line for the analytic model was obtained by generating an IMS model and manipulating the CPU service time and the multiprogramming level to obtain 2 second response time for the desired range of residency times.

As shown by following chart, there is a remarkable comparison of the Resource Demand Factor to actual modeling predictions. The curves start to differ near the "knee" of the curve. This coincides with conditions where operation is on the brink of disaster, where a small change in residency time produces a dramatic change in response time. This is an area of operation which would normally be avoided!

One would not expect exact comparison, since a relatively simple formula could not encompass all of the complexities of an analytic model. However, over the interval where most systems operate, there is a very high degree of conformity.



(1) Information Management System (IMS), A Program Product of IBM Corporation.

(2) Quick-Fetch, A Program Product of DUQUESNE SYSTEMS.

(3) Virtual Fetch, A Program Product of IBM Corporation.

(4) EDP Performance Review, (July, 1986). When Is Chargeback Counterproductive?, David F. Stevens

(5) Lazowska, E. D., Zahorjan, J., Graham, G. S., Sevcik, K. C., "Quantitative System Performance", Prentice-Hall, Inc.

Workload Characterization Using UNIX System Accounting

J. Jerik Tornheim
Richard T. Williams
University of Massachusetts / Boston
BGS Systems, Inc.

ABSTRACT

A UNIX* process is, roughly speaking, a command issued by a user. Berkeley UNIX cuts an accounting record that reports resources consumed by each process. The Berkeley UNIX system accounting utility **sa** reads the accounting records, accregating on either a per user or per command basis. But **sa** output is too fine grained (even after aggregation and sorting) to provide useful numbers for capacity planning. To plan effectively, processes should be combined into workloads, so that cumulative resource use can be analyzed and performance measurements calculated.

We have built **sage** (**sa**, greatly **e**nhanced) to provide an easy method for grouping processes into workloads. **sage** reads a workload specification file provided by the user, to put commands into command-groups and users into user-groups. The user-group and command-group are concatenated to form workload names. Then the existing **sa** routines aggregate useage data for workloads, and sort and display the data. The information that **sage** produces is not available from any other Berkeley UNIX measurement or accounting utility. Preliminary capacity planning decisions can be made using just **sage** outputs and a calculator.

1. sa (Berkeley UNIX System Accounting)

To do any elementary Capacity Planning, you need measurement or accounting data about consumers of service. When UNIX accounting is turned on, an accounting record is cut for each process when it has finished executing [1]. The accounting record consists of fields containing:

- the command name
- the userid
- the groupid
- user-time (time the CPU spent operating in user mode)
- system-time (time the CPU spent operating in kernel mode)
- the beginning time (in seconds since midnight, January 1,1970)
- the elapsed time in seconds
- the CPU storage integral
- the number of I/O operations

Because accounting records are cut only when accounting is turned on, you can report system usage for particular time periods (for example, peak hours) only by turning accounitng on and off, or by filtering the accounting records.

The Berkeley UNIX system accounting utility, **sa**, is the only utility that aggregates these accounting records [2]. **sa** provides many different options for sorting and printing, but because it aggregates based only on userids or on command names, it buries you in numbers. The typical **sa** output contains the option-specified usage figures for each userid or each command, with as many lines in the report as there are userids or commands. Because there may be many hundreds of userids or command names, it becomes very tedious to get useful capacity planning information from **sa**.

sa also provides a 'u' option that reports on each individual process. This report could easily span thousands of lines. When it comes to seeing usage reporting on userids and command names, **sa** ouput is either mutually exclusive or all-inclusive.

2. PROCESSES AS TRANSACTIONS

In capacity planning, the base unit for quantifying workloads is a transaction. There are several inherent shortcomings in using UNIX processes as transactions, or base units of measurement. The first follows from the fact that much work in UNIX is done by forked processes. Resources consumed by forked processes should be charged to the original user-initiated process, but that information is not available in the accounting record. For example consider a simple **make** to compile and link two files together to a single executable

* Trademark of A T & T Bell Laboratories

program. The **make** accounting record may indicate something on the order of 0.05 CPU seconds and 60 elapsed (real time) seconds. But the account records for the two compilations show that each used 5 CPU seconds and 30 elapsed seconds. There is no way to know that the two compiles were forked from the single **make**. Any predictions on future resource consumption based on increased number of **make**s would be significantly underestimated unless the number of compilations was somehow increased, too. A possible solution is to rewrite the Berkeley UNIX accounting utility to include an additional command name field to contain the original user-initiated command.

The second shortcoming involves measuring a baseline response time. The calculated response time for the above **make** might be reasonably close, even though most of the work contributing to the response time is not charged to the **make**. On the other hand, edit commands do get all the work charged to that command. But editing response time should indicate how long it takes the screen to refresh following an editing operation. Because there is a single accounting record for the entire editing session, there is presently no way to determine that value. If an editor were instrumented so that it cut an accounting record for each editing operation, effective response time for editing sessions could be calculated. (Perhaps programmers who design editors should keep capacity planning in mind when they implement their product. You don't need response times to track resource consumption, but you do need them if you do analytic modeling and want to validate the baseline (measured) system.) Given the present limitations in Berkeley UNIX accounting records, elapsed time can be taken as a fair measure of response time for processes that do not fork and do not wait for user input (such as simple compiles and queries like **ps**).

One of our design decisions was to go as far as we could with existing Berkeley UNIX accounting information. We did not want to generate our own, or access or modify the kernel data structures. Inasmuch as no other accounting information is available from Berkeley UNIX that indicates consumers of service, there are no reliable alternatives to using processes as workload building blocks without changing one of our basic design decisions.

Because we have chosen to use processes, we need to consider useful strategies for aggregating processes into workloads. They can be grouped into user-based workloads, command-based workloads, or a combination of the two. Our examples are appropriate for a university environment, where the program was developed. So, three user-based workloads could possibly consist of faculty, student, and system userids. Or you may separate several users who include extensive graphics as part of their workload. The possibilities are numerous.

The choices for grouping commands are equally numerous. You may specify an editing workload or a compiling workload. You may group all the mail routines, or the shell routines, or all the file management routines.

The most informative method for grouping processes is using a combination of user-groups and command-groups. This way you could report of editing done by faculty, or compiling done by students. The best way to visualize this is as a two-dimensional matrix with columns indexed by command-groups and rows by user-groups:

	editing	compiling	mail	misc
students				
faculty				
system				

As you can see, the number of user-groups and command-groups determines how course or fine the resulting performance figures are. Given this functionality, answers to the following questions will be readily available:

1) How much are the courtesy accounts costing?

2) Who are the big **troff** users?

3) What impact on system resources will result from another CS 342 class.

4) What happens if the CS 650 class switches from a compiled language to an interpretive one?

These are hard questions to answer with **sa**. If asked these questions now, the capacity planner or hardware manager must just guess and hope that the guess will not come back to haunt him or her later.

3. sage (OR sa, GREATLY ENHANCED)

sage enhances **sa** by allowing the user the flexibility described above to direct the aggregation of processes.

sage reads a workload specification file, which includes

both a time interval, and description of the aggregated workloads. The interval specification essentially provides the filtering of accounting records mentioned earlier. We felt that while we were aggregating, we could also check for a specific interval. The aggregation description is actually three sequences of regular expressions, with most of the regular expressions followed by a group name.

The first sequence governs the aggregation of commands into command-groups. A command sequence might be:

```
# put "vi", "ed", and "emacs"
# into an "edit" command -group
/^vi$/              edit
/^ed$/              edit
/^emacs$/           edit
# any command beginning with "cc"
# goes in a "comp" command-group
/^cc/               comp
# any command containing "mail"
# goes in a "mail" command-group
/mail/              mail
# any command not already in a
# command-group goes in "-misc"
/./                 -misc
```

As each accounting record is read, the command name is compared to the first regular expression in the command sequence. If a match is found, that command is put into the command-group whose name appears following the regular expression. If no match is found, the second regular expression is compared, and so on. Group names are maintained by a hash table, which is updated each time a match is found.

In the above example, the last regular expression matches any string. This means that any command not already explicitly grouped is aggregated into this "-misc" command-group. We recommend this practice; if you don't do this, you may find your **sage** output longer than the **sa** output you were trying to condense. This is because you generate a unique command-group for each command not matched by this scan of the command sequence, and therefore a unique workload for each user-group that executes that command.

Thus, briefly, to process an accounting record cut by a **sendmail** command, **sendmail** is first compared with **vi**; there is no match. Then **sendmail** is compared with **ed**; again no match. The match occurs when **sendmail** is compared with /mail/, so **sendmail** is put into the command-group "mail." This command-group name is put into the hash table, using the key "sendmail."

The second sequence determines how users are aggregated into user-groups, using a variation on the above method. An example of a user sequence is:

```
# put operator into a user-group
# called "oper-"
/^operator$/        oper-
# aggregates all users containing
# "uu" in a "uucp" user-group
/^uu/               uucp
# aggregates directories
# containing of only numbers into
# a user-group with "CS" as a
# prefix, followed by that
# directory name
/^[0-9]*$/          CS&
```

The hash table for the user-groups is set up first (rather than on the fly, as with command-groups) by processing the password file as follows:

1) The login name is matched against the user sequence of regular expressions. If a match is found, that user is included in the named user-group.

2) If no match is found at the login name level, the path is traversed upward toward the root. Each directory is now matched against the same sequence. Once again, if a match is found, the user is included in the named user-group. If the sequence is exhausted before a match is made, the path is continued upward and a new search for matches begins.

3) If the user's path is traversed all the way to the root directory with no match found, then the user is included in the default user-group. This default user-group is usually the parent directory. For example, if my path is:

/u/umb/cs/crtsy/jerik

then my default user-group is "crtsy."

4) The third sequence of regular expressions is used to negate default user-groups. An example of this third sequence is:

```
# any directory beginning with
# "spr8", "sum8", or "fal8" is
# not a default user-group
/^spr8/
/^fal8/
/^sum8/
```

Suppose Dick's path is:

/u/umb/cs/678/spr87/rtw

indicating that he is enrolled in class CS678 in the spring of 1987. His user-group should probably be based "678" and not "spr87." If a particular directory is matched with any of the regular expressions in this third

sequence, then that directory is not the default user-group.

5) The key to the hash table is not the user name but the userid, since that's what the accounting record contains.

Once again, using the sample workload specification file, "rtw" does not match any of the expressions. Now "spr87" is checked against them all, also with no match. Because this is the parent directory, we check whether to eliminate this as a default user-group. The first expression in the third sequence matches, so at this point there is no default user-group. Now "678" is compared with the second sequence, and a match is found with "/^[0-9]*$/" with the result that he is assigned into the "CS678" user-group. Because a match was found with the second sequence, no default user-group is used, although in this case, "678" is his default user-group (because "spr87" is not).

In addition, our example demonstrates how command and user group specifications use the ampersand character ('&'): if the corresponding group name contains an ampersand, the ampersand is replaced by the string just matched (either the command name or the user/directory name, depending on the context).

Most user paths are designed so that you would naturally want to aggregate users based on common ancestors. Consequently, user names are often not included in the workload specification file, since ancestor names are sufficient for effective aggregation.

When an accounting record is read, the process' user-group name and command-group name determined as above are concatenated and truncated to ten characters to form the workload name. The resource usage figures of the process are aggregated under this workload name. Then with only small changes, we let **sa** sort and print the output into manageable and useful workloads that have been specified by the user.

The 'g' option of **sage** reads in this workload specification file and produces a list of all the user-groups with all the login names in each group. A list of all the command-groups containing specified command names is also produced. By modifying the workload specification file, and then running **sage -g**, the capacity planner can determine how the users and the specified commands are grouped before processing the accounting records.

The 'w' option provides the same output, but first processes all the accounting records. This way, all the processed commands will be listed with the appropriate command group.

4. sage OUTPUT

We have included examples of **sa** and **sage** output to demonstrate the benefits of using **sage**. Appendix A is the output from **sa -m**, based on login names. Appendix B is the output from **sa**, based on commands. Appendix C is the output from **sage**, based on workloads. We did not change **sa**'s internal structures, sorting routines, and numerical aggregation, and made only minor changes in the output options. We have designed **sage** so as to change normal **sa** output routines as little as possible. Because **sa** outputs are usually very long, column headers are not good field labels. In **sa** output each variable has a suffix which defines its value. For example the "l" option produces lines with the format:

```
17146.06re    37.06u    37.06s    27avio    74k
```

(In this example, we do not show the first and last columns, which are the number of individual processes and the command name.) The suffixes for the shown (middle) five columns are (in order):

1) real (or elapsed) time
2) user time
3) system time
4) average number of I/O operations per process
5) cpu-time average core usage

These suffixes make a busy report. For short reports headers would be better. So we have added an "h" option, which only displays the suffixes on the first line and suppresses them on the rest.

sa does not provide the beginning time of a process. We discussed response time and forking questions above. We see no resolution, so we provide an 'e' option, which displays the complete contents of the accounting record, including the start time. With this information, it may be easier to track forked processes. The 'e' option disables all aggregation.

A possible future post-processor might produce the two-dimensional matrix shown earlier with appropriate values in each box, or input to a spreadsheet, database, or graphics package.

5. SUMMARY

To do elementary capacity planning, it is necessary to quantify the types of work done on a system. If arrival rates and service time at the CPU and disks can be measured or calculated, then it is possible to derive utilizations, response times, queue length, and other values necessary for capacity management. Berkeley UNIX presently lacks a utility or method to specify types of processing. In order to obtain resource consumption figures for that processing, **sage** provides this functionality. Consequently, for small systems, **sage** output and a calculator is sufficient to do elementary capacity planning.

For systems with an extended configuration or critical performance criteria, a more sophisticated method of analysis is needed. Capacity planners can construct an analytical model of their system which can then be evaluated using a queueing theory software package.

sage is presently being used at the University of Massachusetts. Several professors who do consulting work at other companies intend to use **sage** at these other sites. **sage** source code will be available to interested users.

We have included the proposed UNIX Programmer's Manual page for **sage** as Appendix D.

ACKNOWLEDGEMENTS

We would like to express our appreciation for the feedback from members of the Computer Science Department of UMass/Boston. In particular, we are indebted to Professor Ethan Bolker. Without his help, guidance, and good humor, this project (and paper) would not have been possible.

REFERENCES

[1] UNIX Programmer's Manual Reference Guide, 4.2 BSD, March 1984, acct(8). Computer Science Division, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley,CA.

[2] UNIX Programmer's Manual Reference Guide, 4.2 BSD, March 1984, sa(8). Computer Science Division, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley,CA.

APPENDIX A (sa -m output)

root	2356	67.36cpu	29126tic	233558k*sec
daemon	515	35.43cpu	8910tio	170630k*sec
UUCP	301	5.33cpu	6636tio	23794k*sec
kathy	70	4.06cpu	1152tio	30267k*sec
godfrey	24	0.53cpu	619tio	4952k*sec
ram	48	0.93cpu	903tio	4709k*sec
eoneil	16	0.23cpu	160tio	920k*sec
bill	74	0.76cpu	800tio	3699k*sec
kwn	36	0.56cpu	693tio	4695k*sec
poneil	2	0.05cpu	18tio	179k*sec
ar	15	0.18cpu	136tio	727k*sec
gputnam	5043	241.87cpu	47192tio	7068710k*sec
sched	36	1.99cpu	364tic	43496k*sec
hung	29	0.37cpu	302tio	1020k*sec
jss	58	2.42cpu	762tio	17735k*sec
clmao	22	1.47cpu	496tio	14157k*sec
gmarsh	168	10.39cpu	2848tio	84645k*sec
rjv	75	4.79cpu	594tio	20682k*sec
chenj	13	0.19cpu	123tio	762k*sec
rtw	444	40.51cpu	10266tio	369687k*sec
rwood	822	39.57cpu	12164tio	289590k*sec
jusong	12	0.13cpu	122tio	501k*sec
kong	12	0.10cpu	49tio	195k*sec
dmclaug	8	0.12cpu	94tio	235k*sec
miki	20	0.69cpu	159tio	2359k*sec
mhos	8	0.15cpu	125tio	629k*sec
ychen	12	0.09cpu	56tio	183k*sec
jyuan	101	2.20cpu	642tio	12165k*sec
speed	21	1.58cpu	198tio	5948k*sec
trinh	42	0.48cpu	229tio	1690k*sec
whojr	102	5.02cpu	1561tio	20389k*sec
klee	22	0.57cpu	316tio	3185k*sec
safari	13	0.18cpu	168tio	582k*sec
gerald	219	3.92cpu	2692tio	67517k*sec
mefitz	300	19.59cpu	5242tio	171251k*sec
yclau	48	1.02cpu	488tio	4720k*sec
jaleh	113	9.44cpu	1991tio	116915k*sec
ygohary	597	19.54cpu	4420tio	319226k*sec
murphy	54	1.32cpu	350tio	10842k*sec
chevy	25	1.67cpu	555tio	36810k*sec
kenm	32	0.59cpu	406tio	3868k*sec
merriam	346	3.02cpu	2046tio	15762k*sec
javed	43	0.43cpu	327tio	1630k*sec

APPENDIX B (sa output - no options)

```
12077   24178.57re     528.78cp      12avio     288k
   37      96.24re      71.02cp      42avio    1318k  diff
  384    1430.90re      62.44cp      40avio     157k  vi
 1075     137.16re      55.49cp       6avio     141k  slint
   60    2420.84re      35.11cp      14avio      37k  rlogind
   68     147.37re      33.67cp      17avio     252k  ccom
   74     121.83re      28.49cp      50avio     155k  troff
  311    7706.42re      25.64cp      43avio     113k  csh
   21     177.35re      15.75cp     670avio      85k  ***other
   56      54.40re      13.48cp      48avio     143k  as
   46      64.05re      11.33cp       7avio     197k  lint1
 1196      40.28re      10.83cp       3avio      17k  cat
   38      29.56re       9.81cp      44avio     118k  sage
   14      22.38re       9.71cp      89avio     132k  yacc
    5     115.93re       9.57cp     370avio      84k  pmdf-maste
  115      75.42re       9.50cp      40avio      88k  cpp
   70      71.65re       8.03cp      21avio     104k  icat.s
  142      88.01re       7.80cp      23avio      44k  ips
   65      55.74re       7.71cp      10avio     104k  ipf.s
    7     438.53re       6.53cp     141avio     585k  lisp
   70      31.79re       6.31cp      20avio      78k  dviimp
   10      37.70re       4.86cp      34avio     401k  latex.real
  383      11.29re       4.57cp       5avio      50k  ls
   26      12.09re       4.35cp      36avio     129k  nroff
   36      24.76re       4.34cp      25avio      18k  grep
   38       8.65re       4.11cp     144avio     112k  ld
 1033    1940.85re       3.91cp       1avio      31k  sh
   70       7.46re       3.81cp      18avio      54k  catdvi
   74     129.57re       3.54cp      12avio     101k  itroff
   40    1072.11re       3.23cp      73avio      81k  uucico
    5      63.52re       2.86cp       5avio     103k  talk
   19     453.70re       2.84cp       5avio      55k  rlogin
    2       6.31re       2.56cp      53avio     182k  lint2
   19     452.87re       2.53cp       0avio      60k  rlogin*
 1060       8.07re       2.46cp       8avio       8k  cp
  138     141.46re       1.95cp      21avio      78k  lpd*
  120     122.74re       1.89cp      30avio      41k  lpr
    6      22.29re       1.85cp      18avio      71k  kermit
  107     385.39re       1.82cp       8avio      49k  make
   47      15.62re       1.58cp      13avio      17k  mo
   13       7.02re       1.48cp      56avio      10k  wc
   30       3.43re       1.40cp      16avio      56k  w
  122      69.47re       1.33cp       4avio      26k  more
 1163       5.46re       1.32cp       3avio       5k  rm
    4       6.33re       1.31cp     730avio      67k  nfxmit
   23       3.89re       1.27cp     110avio     286k  sendmail
   27      11.47re       1.15cp      15avio      64k  spell
   63      30.88re       1.06cp      23avio      69k  mail
   19       6.42re       1.04cp      10avio      52k  sort
    7       5.13re       0.88cp      40avio      96k  dvi-imagen
   41       2.72re       0.88cp       8avio      85k  ps
    7       9.24re       0.79cp      24avio     104k  idvi.s
   71       1.25re       0.77cp       4avio      24k  pwd
    2      26.27re       0.76cp      17avio     184k  rogue
  105       2.22re       0.68cp      14avio      23k  dmesg
    4      15.34re       0.65cp       0avio      45k  kermit*
   85       1.13re       0.65cp       6avio      25k  quota
   70       3.07re       0.63cp       9avio      22k  atrun
   32      12.68re       0.62cp      27avio      16k  man
   10     108.10re       0.60cp      14avio      42k  login
```

APPENDIX C (sage output - no options)

```
12077   24178.57re      528.78cp        12avio      288k
    1       0.40re        0.01cp        33avio       65k   ***other
  227    1089.71re        6.09cp        11avio       69k   cnslt_misc
    3      25.23re        0.45cp        33avio      145k   cnsltedit
    2       0.02re        0.01cp         6avio       32k   cnsltmail
  832    1780.16re       38.50cp        14avio      162k   crtsy_misc
   50      31.87re        0.17cp         6avio       14k   crtsycomp
   90     159.91re        7.20cp        28avio      155k   crtsyedit
   13       2.95re        0.19cp        32avio      189k   crtsymail
 1049    3716.81re       17.72cp         7avio      253k   cs651_misc
  130     877.31re       24.70cp        41avio      163k   cs651edit
   26      12.12re        0.25cp        20avio      116k   cs651mail
   63     133.01re        0.78cp         8avio       58k   cs656_misc
   44      86.88re        0.34cp         8avio       56k   cs680_misc
  824    2290.91re       20.84cp        10avio      144k   ops_misc
  103     270.46re       12.28cp        45avio      139k   opsedit
    5       1.04re        0.09cp        36avio      203k   opsmail
 2975    8794.37re       88.49cp        13avio       63k   sys__misc
   84       3.85re        0.88cp        21avio      274k   sys_mail
  116    1076.54re        3.08cp        40avio       76k   sys_system
 5508    3702.66re      208.96cp         9avio      540k   usr_misc
   49     199.52re        0.21cp         9avio       13k   usrcomp
   87     186.23re        8.45cp        40avio      185k   usredit
   55      20.78re        0.97cp        40avio      195k   usrmail
```

APPENDIX D

SAGE(8) UNIX Programmmer's Manual SAGE(8)

NAME

sage - system accounting (sa, greatly enhanced)

SYNOPSIS

/etc/sage [-abcdDefghjkKlmnrtuvw]
[-G groupfile] [-P passwdfile]
[-S wklsummaryfile] [-W wklspecfile]
[rawacctfile]

DESCRIPTION

sage condenses accounting information by user-specified workloads and gives aggregate reports on the workloads, a la sa(8), over a specified time interval.

If a file name is given as the last argument, that file will be treated as the accounting file; /usr/adm/acct is the default.

sage takes input of two types. One is system accounting data, as accumulated by accton, in the file /usr/adm/acct. The other is workload grouping descriptions, specified in the file /usr/adm/wklspec. This file consists of four parts. The first part contains information about the time interval you are studying. The second part contains a sequence of regular expressions which are used to group commands into command-groups. The third part contains a sequence of regular expressions used to group login names into user-groups. The fourth part contains a sequence of regular expressions which are used to eliminate particular default user-groups. The division between the first and subsequent parts of the wklspec file are lines which start with an equal sign ('='). Blank lines and lines beginning with a number sign ('#') are ignored.

The first part of the wklspec file contains the starting time of the interval that you will be studying, in 24 hour format, and the length of the interval, in minutes. For example: 10:00 30 indicates an interval of thirty minutes, starting at 10:00 am. 15:15 45 indicates an interval of 45 minutes, starting at 3:15 pm. It is assumed (rather pretentiously) that accounting data for one day only is stored in the input file /adm/acct. To insure that, accton should be run from a shell script that restricts its duration.

Each regular expression in the second and the third part is followed by a group name. As each accounting record is processed, the command name is compared to each regular expression in the second sequence until a match is found. If a match is found, that command name is put into the command-group whose name appears following the regular expression. An example of a command sequence is:

```
# put vi, emacs, and ed into
# an "edit" workload
/^vi$/      edit
/^emacs$/   edit
/^ed$/      edit
# put all commands containing "mail"
# into a "mail" workload
/mail/      mail
# put everything else
# into a "-misc" workload
/./         -misc
```

Before processing any accounting records, all login names in /etc/passwd are put into user-groups. Each login name is compared to each regular expression in the third sequence until a match is found. If a match is found, the corresponding userid is put into the user-group whose name appears following the regular expression. If no match is found, the path is traversed up to the root, with each directory compared against the same sequence. An example of a user sequence is:

```
# put "operator" into an "oper-"  workload
/^operator$/   oper-
# put all directories with only numbers into
# a workload beginning with "CS"
/^[0-9]$/      CS&
```

If there is no match found with the third sequence, a default user-group is used. This is the lowest directory name which does not match any regular expression in the fourth sequence. An example of a default sequence is:

```
# any directory beginning with "spr8" is not
# a default user-group
/^spr8/
# "crtsy" is not a default user-group
/^crtsy$/
```

sage output fields are those of sa, except the obvious: sage prints workload names where sa printed commands or login names. Some fields are labeled:

"cpu"	for the sum of user and system time (in minutes),
"re"	for real time (also in minutes),
"k"	for cpu-time averaged core usage (in 1k units),
"avio"	for average number of i/o operations per execution.

With options, fields labeled:

"tio"	for total i/o operations
"k*sec"	for cpu storage integral (kilo-core seconds),
"u"	for user cpu-time alone (in minutes),
"s"	for system cpu-time alone (in minutes).

sage retains most of sa's googol of options, and adds a few dozen more:

a Print all workload names, even those containing unprintable characters and those used only once. By default, those are placed under the name '***other'.

b Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system time.

c Besides total user, system, and real time for each workload, print percentage of total time over all workloads.

d Sort by average number of i/o operations.

D Print and sort by total number of disk i/o operations.

e Superseding all other flags, print the entire contents of each accounting record, preceded by the appropriate workload name and login name. Aggregation is disabled.

f Force no interactive threshold compression with -v flag.

g Without processing the account records, read the wklspec file and print out the login names for each user-group and the specified commands for each command-group.

h Suppress the suffix description for each value except on the first line (the "h" stands for "header").

j Instead of total minutes for each workload, give seconds per call.

k Sort by cpu-time average memory usage.

K Print and sort by cpu-storage integral.

l Separate system and user time; normally they are combined.

m Print number of processes and number of CPU minutes for each workload.

n Sort by number of calls

r Reverse order of sort

t For each workload report ratio of real time to the sum of user and system times.

u Superceding all other flags (except the "e" flag), print for each process in the accounting file the userid, cpu seconds, cpu storage integral, i/o operations, and command name.

v Followed by a number n, types the name of each workload referenced n times or fewer. Await a reply from the terminal; if it begins with 'y', add the workload to the category 'junk'. This is used to strip out garbage.

w After processing the account records, print out the login names for each user-group and the commands for each command-group.

G The following filename is used as the user-group and command-group listing file instead of /usr/adm/wklgroup.

P The following filename is used instead of /etc/passwd as the system password file.

S The following filename is used instead of /usr/adm/wklacct as the workload summary file.

W The following filename is used instead of /usr/adm/wklspec as the workload specification file.

FILES

/usr/adm/acct	raw accounting
/usr/adm/wklgroup	user-group and command-group listing
/usr/adm/wklspec	workload specification
/usr/adm/wklacct	per-workload summary
/etc/passwd	system password file

SEE ALSO

ac(8), acct(2), sa(8)

BUGS

The number of options to the program is absurd.

A BENCHMARK COMPARISON OF DB2 AND THE DBC/1012

G.C. Steindel and H.G. Madison
Great-West Life Assurance Company

ABSTRACT

Over time Information Services departments have attempted, in a timely manner, to provide adaptable, maintainable, and cost effective applications to support the information requirements of their organizations . From this need and desire have evolved the use of shared data-base management systems(DBMSs) and the techniques of data modeling and normalization. While providing adequate performance, DBMS's based on the hierarchical and network models have failed to achieve acceptable levels of application adaptability, maintainability, and ease of use. The relatively recent introduction of DBMS's based on the relational model and the use of entity-relationship data modeling techniques however, appear to hold promise for addressing these shortcomings, though at a possible detriment to performance.

The results oriented nature of relational data-bases(RDBs) which require the specification of the desired results rather than the method for attaining the results has facilitated the implementation of these systems in specialized hardware in addition to the more traditional software approach. The relational data-base machine approach has the potential to offer improved price-performance over software implementations on general purpose hardware and is of extreme interest considering the general concern over responsiveness of relational systems.

This paper presents an initial comparison of these two approaches to providing relational services by contrasting the cost and performance of IBM's DB2 and Teradata Corporation's DBC/1012 through the use of a benchmark.

1.0 INTRODUCTION.

Great-West Life is a large Canadian insurance company utilizing primarily IBM hardware and software. Most applications have been developed in-house using IMS technology. Although this technology for the most part has allowed the company to accomplish its objectives it has been highly labour intensive in data-base design, as well as application maintenance. While providing the necessary basis for the day to day administrative tasks of the company these applications have not effectively addressed the need for the accessibility of management information that was assumed to be available with data-base technology.

The advent of relational data-base technology appeared to address many of the problems which Information Systems departments had been having with IBM's hierarchical data-base approach. Data-base design efforts would be greatly reduced by allowing normalized entity-relationship data models to be mapped directly into relational tables. Adaptability of both applications and data models would be enhanced by the ability to readily add new tables and entities without disrupting existing applications through the use of techniques such as views. Accessibility of data would be improved by providing the more business intuitive paradigm of the table, and removing the responsibility for data-base navigation from non-technical people attempting to find information. These benefits although laudable and necessary in the business community would likely not come free. Rumors still run rampant today about the ability of relational data-base systems to consume general-purpose processors with the greatest of ease. Since relational systems are based on the principle of describing the results required rather than the method necessary to attain the results, they by nature have a straightforward and highly structured language as their interface. This has opened the possibility of developing special purpose hardware for the delivery of relational services. The incentive for these sorts of developments are oriented primarily to providing higher levels of performance and reduced cost relative to conventional software based systems.

Over the last year Great-West Life, like many other IBM installations, installed IBM's Data Base 2 (DB2) [1] relational product and experimented with it in a variety of test applications which were primarily of a decision support nature. During this investigation period we encountered several situations where staff were readily able to formulate queries which would consume what the data centre staff felt were inordinate amounts of processor resources. These situations heightened our awareness of the potential performance problems that relational processing might bring if adopted in a full scale production environment , and subsequently prompted us to investigate other means of providing relational services. In mid 1986 we became aware of a relational data-base machine product offered by Teradata Corporation of Los Angeles, California which appeared to fit reasonably well into our IBM mainframe environment. In order to evaluate the Teradata product known as the DBC/1012 [2] we decided to conduct a benchmark comparison between it and DB2. The remainder of this paper will discuss the objectives, environment, results and conclusion of the benchmark.

2.0 BENCHMARK OBJECTIVES.

While still being fairly new at the DB2 game the first objective of the benchmark was to determine just how far we could push our 3090-200 system with query workloads. We had seen several instances where complex queries had used large amounts of processor resources but were not sure just how many of these it would take to saturate our system. We anticipated that this was an item of concern since the accessibility which the relational approach afforded the owner of a database would increase the demand for large database inventories and management reporting. This type of workload would be primarily of a batch nature in the off shift. However, if our interactive query experience was any indicator, it would be considerably more processor intensive than had been our previous experience with other non-relational over night batch processing. The primary orientation of this part of the benchmark was toward throughput, and capacity of a system.

The second objective of the benchmark was to compare the performance of an interactive query workload. Here the factors of interest were primarily elapsed time and processor resources consumed which is the primary element of cost to the requestor.

While up to this point Great-West Life's usage of relational processing was focused on query type applications aimed at providing improved management reporting, we realized that our benchmarking activities would have to cover the area of transaction processing in order to provide a relatively complete picture of the products' capabilities. It was anticipated that over time the existing IMS applications would begin to include relational components as they were maintained or redeveloped. Again, the measures of responsiveness and resource consumption were of prime interest. While the issue of throughput was quite important to us in investigating transaction oriented systems we were unable to delve into it due to the significant effort that would have been necessary to build transaction drivers that were portable to the Teradata benchmark facilities.

Lastly, any additional information that could be gleaned about utility, data-base load processing, and index creation performance would be considered gravy.

3.0 TERADATA DBC/1012 SYSTEM OVERVIEW.

The DBC/1012 is a special purpose computer specifically designed to execute SQL relational data-base queries and updates generated from a general purpose computer. The general purpose host runs a special application called the Teradata Director Program (TDP) [3] which is used to interface between the DBC/1012 and the host applications requesting relational services. The TDP receives SQL requests from the various application programs running in either a MVS or VM host. These requests are then transmitted to the DBC/1012 across a standard 370 block multiplexer channel . Each channel is connected to an Interface Processor (IFP) on the data-base machine as is shown in Figure 1. The IFP is responsible for parsing the SQL request and developing the execution plan which it then schedules on the Access Module Processors (AMP's). Attached to each AMP are one or two Disk Storage Units (DSU's) on which the tables are stored. All AMP's and IFP's are connected together with a branching tree-like device called the Y-net shown in Figure 2. The Y-net is the part of the DBC/1012 which facilitates the broadcasting of requests to, and merging of results from the AMP's which operate in parallel to produce a relational result. When the AMP's have completed the execution of an SQL statement the results are transmitted to the IFP which send them back up the 370 channel to the TDP and subsequently to the appropriate host application.

AMP's using a hashing technique. If an IFP determines that a row of a table can be accessed by its primary key, the key is hashed to determine the appropriate AMP and followed by a single AMP disk search to locate the row. The result row is then returned up the Y-net to the IFP. If an IFP determines that the result rows reside on more than one AMP the request is sent up the Y-net toward the root, turned around, and broadcast simultaneously to all AMP's. Each AMP then performs the appropriate disk search and simultaneously present their results to the Y-net. The Y-net then performs a balanced-line merge, continually selecting the result row with the lowest key value from the AMP's until all result rows are transmitted to the IFP.

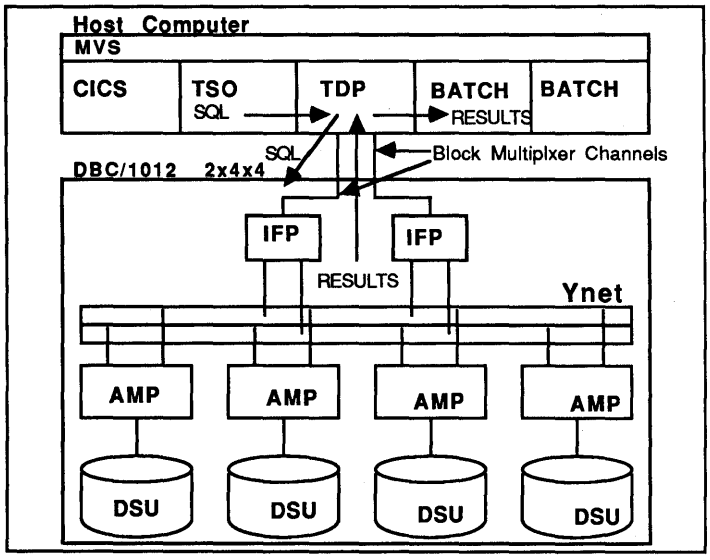


Figure 1. MVS Host - DBC/1012 Configuration

For each non-unique secondary index (or non-unique primary index) defined on a base table, an additional table is stored on each AMP [4]. These secondary index tables contain the secondary key values for the base rows on their own AMP. The secondary index also contains one or more track addresses of the appropriate blocks containing base rows of each non-unique key. Unique secondary indices are treated as a special case of uniquely keyed base tables and are accessed by hashing the secondary key valued to select an AMP which then determines the corresponding primary key value which is hashed to determine the AMP containing the base table row.

The larger the number of AMP's attached to a DBC the more parallelism that is incorporated into table searches, and the faster the response to a data-base request. Teradata has benchmarks which show that query response times decrease linearly with the increase in the number of AMP's within a tolerance of approximately 4% error.

The Teradata Query Language (TEQUEL) which is used to manipulate data on the DBC/1012 is very similar to IBM's Structured Query Language (SQL). In addition to being able to code TEQUEL in various third level language programs, which is translated by a language preprocessor, host utilities are provided for the direct submission of TEQUEL to the DBC/1012 via the TDP.

4.0 BENCHMARK ENVIRONMENTS.

4.1 DB2.

The DB2 portions of the benchmarks were run initially on a dedicated 3090-200 with 64 megabytes of real storage running the MVS/XA operating system at level 2.1.3. The DB2 software was release 1.2 The tables used in the study were loaded on 8 volumes of single density 3380 DASD which were distributed across 8 internal 3380 paths, 3 3880 controllers with 2 channel paths to each controller. Table 1. shows details of the various tables used in the benchmark. The tables were stored one per tablespace, in tablespaces that had been defined with the CLOSE(NO) option. Default values were accepted for the FREESPACE and FREEPAGES parameters. None of the tablespaces were partitioned. The RUNSTATS operation was performed for all of the tablespaces.

The sizes indicated are data sizes as opposed to allocated DASD space. The total DASD space actually allocated on the 8 volumes was approximately 3 gigabytes. The DB2 buffer pool BP0 was allocated at 24 megabytes, and 12 megabytes of BP1 were used for index buffers. In this environment we moved tables around until we achieved a reasonably balanced system with all DASD devices averaging 46.8% device active and two devices running at 86% device active over the peak 1 hour period in the batch query workload. During this 1 hour period the processor ran at 78% busy. Throughout all of the benchmark runs essentially no paging activity was measurable. As in many data centres where equipment configurations rarely remain constant, the 3090-200 processor was upgraded to a 3090-400 with 128 megabytes of real memory and 128 megabytes of expanded memory. This upgrade took place after the initial DASD balancing was completed. All the remaining data reported in this paper about DB2 CPU usage were derived from this final configuration.

4.2 Teradata DBC/1012

The benchmarks were run on a 8x24x24 DBC/1012 (8 IFP's, 24 AMP's, and 24 372 megabyte DSU's) connected to an Amdahl 5860 with the Multiple Domain Feature (MDF). The DBC/1012 was running the 042 level software with TDP release 2.3 running under MVS/SP 1.3. The MDF partition was assigned 15 megabytes of real memory and TARGET and MAX processor allocations of 20%. A second partition was running a VM system with TARGET and MAX processor allocations of 100% and 80% respectively. The equivalent indices as were used in the DB2 test were defined for the DBC/1012 benchmark runs.

4.3 Charging and Rate Setting

In companies which provide data processing services to their staff through internally operated facilities which employ a chargeback mechanism, it is often not the overall cost of providing the services to the whole organization that is of most interest, as are the charges which are billed to the individual users of the services. When conducting the benchmark it was important to be able to provide the users a comparison of the rates they could expect in each of the environments.

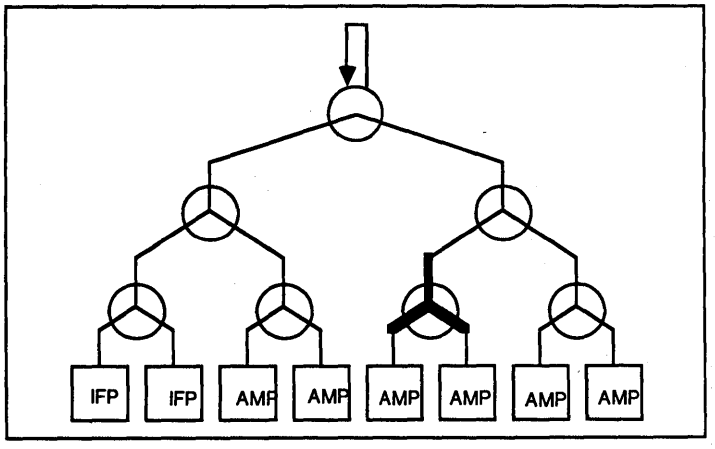


Figure 2. Conceptual view of the Y-Net

	Table Name	Table Description	Row Size in Bytes	Key Size in Bytes	000's of Rows	Table Space MB	Index Space MB	Num. of Ind's
1	APCO	Appl'd Coverage	24	18	160	3.8	2.9	1
2	BLAR	Policy Billing Arr.	114	15	615	70.1	9.2	1
3	BRAN	Br. Translation Tab.	50	5	1	0.1	0.0	1
4	CAGD	Commission Share	28	18	120	3.4	2.2	1
5	CAIN	Covg.-Agent Info	106	26	1,185	125.6	30.8	1
6	CCMD	Covg.-Agent Comp.	183	31	2,070	378.8	64.2	1
7	CCTT	Covg. Code Trans.	15	5	1	0.0	0.0	1
8	CIAR	Covg. Insured Addr.	18	18	1,410	25.4	25.4	2
9	CLAD	Client Data - Addrs	242	8	1,330	321.9	10.6	1
10	CLBA	Client Data - Basic	149	6	1,275	190.0	7.7	1
11	CMAC	Commission Acct	155	6	15	2.3	0.1	1
12	COIN	Coverage Insureds	26	16	1,410	36.7	22.6	2
13	COVR	Coverage Info	236	16	1,595	376.4	25.5	6
14	EVEN	Issue/Undwtg Events	247	12	850	210.0	10.2	4
15	IFPA	Insured for Package	35	18	145	5.1	2.6	2
16	INAP	Insured Appl. Pkg.	37	12	135	5.0	1.6	4
17	OREQ	Outstanding Req's	67	12	135	9.0	1.6	2
18	PAIN	Policy-Agent Info	25	20	560	14.0	11.2	1
19	PLCY	Policy + Appl Info.	343	10	585	200.7	5.9	6
20	PLON	Policy Owner Table	23	18	590	13.6	10.6	2
21	REPL	Replacement Policies	28	28	5	0.1	0.1	2
22	RQFN	Request Form Num.	39	39	115	4.5	4.5	1
23	RQPS	Req'd Policy Suppl.	32	22	5	0.2	0.1	1
	Totals				14312	1996.5	249.6	45

Table 1. Table and Index Descriptions

Great-West Life sets chargeback prices by projecting the annual cost for providing a resource such as a processor, and dividing it by the projected annual (chargeable) utilization of the resource. Utilization of resources by the data centre staff is considered non-chargeable The annual cost of a resource includes its depreciation, maintenance, cost of capital, floor space, environmental requirements, and required support staff such as operators and systems programmers. In the case of resources such as processors, other indirect costs are included such as those for disaster recovery, work scheduling, network and help-desk operation, and the provision of technical support to the user community.

In the case of the DBC/1012 while it was straightforward to estimate its direct costs, it was rather difficult to estimate the indirect costs and the projected usage of the resource. On the cost side, to maintain uniformity with the DB2 environment, the direct cost were loaded with the same proportion of indirect costs as were experienced for the DB2 host processors. As for the projected usage, an interesting phenomenon exists at Great-West Life. While the processing capacity of the data centre has grown consistently and dramatically over the past 5 years from a 3081 model D and a 3033 MP to a 3090 model 200 and 400, the chargeable utilization within any one year stays approximately constant at 20% of the theoretical maximum utilization of the installed processors. In estimating the price that would be charged for the utilization of a DBC/1012 the projected utilization of a processor was assumed to be 20% of its theoretical maximum.

In order to encourage the use of offshift capacity, Great-West Life offers a 50% price reduction for the use of processor resources outside of it 8 A.M. to 5 P.M. week-day prime usage period. The current situation is that 53% of all processor utilization takes place in the prime usage period with the remaining 47% taking place outside of the prime period. Again, the same assumptions were made in the determination of a DBC/1012 usage price. Although the DBC/1012 contains both SQL processing capabilities and storage capabilities, only the costs associated with the processor portion of the DBC/1012 were considered to make comparisons to the DB2 host environment credible.

Application of these pricing and projection policies resulted in a prime shift rate of $2160 for one hour of TCB and SRB CPU time as recorded by SMF ($1080 outside of prime shift) for the 3090-400, and $1995 and $1162 for one hour of processing time per processor on a 8X24X24 and 4X8X8 DBC/1012 respectively. The DBC/1012 maintains a separate set of system tables in which it accumulates the processor utilization necessary to operate this type of charging. These table can be can be queried and reset by the installation as necessary.

5.0 BENCHMARK RESULTS.

5.1 BATCH QUERY WORKLOAD.

The batch query workload was a selection of 13 intentionally long running complex queries which produced a relatively even distribution of device activity across the 8 3380 volumes used in the DB2 portion of the benchmark. In addition to running all 13 queries concurrently in a stand-alone environment, they were run serially to determine the degradation caused by query contention. Table 2 provides the elapsed time results of the 2 runs of the Batch Query workload. All times are given in seconds with the DB2 CPU times being the sum of the SMF reported TCB and SRB times. Estimated elapsed times are provided for an 8 AMP DBC/1012 configuration and were determined by multiplying the 24 AMP elapsed times by 3 based on Teradata's linearity claims. The host CPU time on the Amdahl system was so small as to be insignificant. Most of the queries, although long running, returned less thai, a 1000 rows which generated very low host consumption since the majority of processing was performed on the DBC/1012.

When compared to the 24 AMP DBC/1012, DB2's elapsed time was 5.9 times as long, and an estimated 2.0 times as long, for an 8 actuator sized DBC/1012 configuration. Running all the queries concurrently increased the DB2 CPU time by 10% over their serial execution, while reducing the total elapsed time for the whole job stream from a little over 19 hours to just under a 11.4 hours or a 68% reduction. The DBC/1012 configuration produced a 23% reduction in elapsed time from 3.3 hours serially down to 2.6 hours when multiprogrammed.

In a multiprogrammed environment an 8 AMP DBC/1012 was outperformed by DB2 by 40% on average, while a 24 AMP processor produces a 1.8 times improvement in responsiveness over DB2.

If one were to remove the 2 queries that DB2 had some difficulty in processing (Q02, and Q06) from the analysis, assuming that better indexing or query tuning could have brought their elapsed times down closer to the rest of the sample, the above relative performance ratios drop into the range of equality.

In order to compare the price performance of the two alternatives a projected chargeback algorithm was derived for the DBC/1012 environment. Similar assumptions were made for both the 3090 and DBC/1012 in developing the cost pools and anticipated usage for calculating the chargeback rate. In the Great-West Life environment a 50% rate reduction is given for the use of resources in the offshift. The resulting rates were applied to the CPU consumption of both systems and shown in Table 4 for an 8 AMP and 24 AMP DBC/1012 configuration respectively. The DBC/1012 was able to process this workload for between 17 and 27 percent of the DB2 cost on a 3090-400 depending on the size of the DBC/1012 configuration. The 24 AMP DBC/1012 resulted in the better of the two price performance percentages since it spread its fixed overhead over a larger usage base.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
	----------DB2----------						---DBC/1012 Elapsed Sec.---					-----DB2/DBC Ratios---			
Run--	CPU Sec.--		(1)/	Elapsed Sec.		(4)/	---24 AMP--------		8 AMP----		(7)/	(4)/	(5)/	(4)/	(5)/
#	Ser.	M.P.	(2)	Ser.	M.P.	(5)	Ser.	M.P.	Ser.	M.P.	(8)	(7)	(8)	(9)	(10)
5	68	79	1.17	296	542	1.8	156	2270	467	6811	14.6	1.9	0.2	0.6	0.1
24	136	299	2.21	306	930	3.0	1121	5816	3362	17448	5.2	0.3	0.2	0.1	0.1
9	74	90	1.21	437	1392	3.2	323	3657	970	10970	11.3	1.4	0.4	0.5	0.1
3	233	288	1.24	447	1229	2.7	336	4222	1007	12665	12.6	1.3	0.3	0.4	0.1
4	160	194	1.21	669	1878	2.8	348	4353	1045	13058	12.5	1.9	0.4	0.6	0.1
26	395	444	1.12	688	1949	2.8	596	5253	1788	15758	8.8	1.2	0.4	0.4	0.1
25	535	581	1.08	2156	4357	2.0	291	3585	873	10756	12.3	7.4	1.2	2.5	0.♠
22	760	883	1.16	2314	4343	1.9	445	4531	1336	13594	10.2	5.2	1.0	1.7	0.3
21	897	1051	1.17	2549	6194	2.4	1613	3606	4838	10819	2.2	1.6	1.7	0.5	0.6
15	1670	1909	1.14	3694	7342	2.0	3048	9536	9143	28609	3.1	1.2	0.8	0.4	0.3
1	2297	2378	1.04	9711	16960	1.7	2543	9204	7629	27612	3.6	3.8	1.8	1.3	0.6
2	1478	1583	1.07	11403	22979	2.0	830	6060	2489	18180	7.3	13.7	3.8	4.6	1.3
6	2404	2423	1.01	34298	40910	1.2	81	1195	244	3586	14.7	422	34.2	140	11.4
Total and Ratios															
	11105	12K	1.10	68967	111K	1.6	12K	63289	35190	189866	5.4	5.9	1.8	2.0	0.6
Totals and Ratios excluding. Query 6															
	8701	9778	1.12	34669	70095	2.0	11649	62093	34946	186280	5.3	3.0	1.1	1.0	0.8
Totals and Ratios excluding. Query 6 and Query 2															
	7223	8195	1.13	23266	47116	2.0	10819	56033	32457	168100	5.2	2.2	0.8	0.7	0.3
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)

Table 2. Batch Query Workload -- Run Statistics and Charges
(Ser.=Serial execution, M.P.=Concurrent execution)

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
	---------On Demand Charges in $--------					--Charges in $ with Deferred Execution--				
Query				(1)/	(1)/				(6)/	(6)/
#	DB2	24 AMP	8 AMP	(2)	(3)	DB2	24 AMP	8 AMP	(7)	(8)
12	1.80	1.50	2.62	0.83	1.46	1.80	1.50	2.62	0.83	1.46
22	4.80	4.53	7.91	0.94	1.65	4.80	4.53	7.91	0.94	1.65
4	0.60	59.30	103.61	98.8	173	0.60	59.30	103.61	98.8	173
28	3.00	7.20	12.59	2.40	4.20	3.00	7.20	12.59	2.40	4.20
30	3.00	94.37	164.90	31.5	55	3.00	94.37	164.90	31.5	55.0
24	7.20	91.76	160.34	12.7	22	7.20	91.76	160.34	12.7	22.3
25	37.80	72.60	126.85	1.92	3.36	37.80	72.60	126.85	1.92	3.36
5	16.20	71.46	124.87	4.41	7.71	16.20	71.46	124.87	4.41	7.71
17	9.00	13.07	22.84	1.45	2.54	9.00	13.07	22.84	1.45	2.54
29	18.00	33.30	58.18	1.85	3.23	18.00	33.30	58.18	1.85	3.23
18	120.60	63.54	111.04	0.53	0.92	120.60	63.54	111.04	0.53	0.92
3	29.40	71.70	125.28	2.44	4.26	29.40	71.70	125.28	2.44	4.26
10	51.00	65.46	114.38	1.28	2.24	51.00	65.46	114.38	1.28	2.24
6	132.60	76.11	132.98	0.57	1.00	132.60	76.11	132.98	0.57	1.00
Subtotal and Ratios (request ~< 10 minutes)										
	435.00	725.89	1268.40	1.67	2.92	435.00	725.89	1268.40	1.67	2.92
26	187.20	156.28	273.07	0.83	1.46	187.20	156.28	273.07	0.83	1.46
11	388.20	210.54	367.89	0.54	0.95	388.20	210.54	367.89	0.54	0.95
9	466.20	173.57	303.29	0.37	0.65	466.20	173.57	303.29	0.37	0.65
23	340.20	407.13	711.40	1.20	2.09	340.20	407.13	711.40	1.20	2.09
8	113.40	46.16	80.65	0.41	0.71	113.40	46.16	80.65	0.41	0.71
16	1176.00	386.05	674.57	0.33	0.57	588.00	386.05	674.57	0.66	1.15
27	354.00	230.90	403.47	0.65	1.14	177.00	230.90	403.47	1.30	2.28
21	579.60	292.21	510.59	0.50	0.88	289.80	292.21	510.59	1.01	1.76
7	1375.80	937.79	1638.66	0.68	1.19	687.90	937.79	819.33	1.36	1.19
15	1404.60	935.76	1635.11	0.67	1.16	702.30	935.76	817.56	1.33	1.16
20	2655.00	1626.64	2842.34	0.61	1.07	1327.50	1626.64	1421.17	1.23	1.07
Subtotal and Ratios (request > 10 minutes)										
	9040.20	5403.01	9441.05	0.60	1.04	5267.70	5403.01	6382.99	1.03	1.21
Totals and Ratios										
	9475.20	6128.90	10709.44	0.65	1.13	5702.70	6128.90	7651.39	1.07	1.34
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Table 3. Interactive Query Workload -- Projected User Charges

5.2 Interactive Query Workload.

The interactive query workload was a sample of 26 queries which had been developed by end-users who had been working in a decision support environment. The queries had been created using the IBM Query Management Facility (QMF) running on-line under the Time Sharing Option (TSO) of Multiple Virtual Storages Extended Architecture (MVS/XA) operating system. A short description of the SQL functions used in the queries and the EXPLAIN results can be found in Appendix A. It can be seen that significant use of indices is made in the execution of the queries.

	(1)	(2)	(3)	(4)	(5)
Run #	DB2	24 AMP DBC	(2)/ (1)	8 AMP DBC	(4)/ (1)
Q05	$20	$20	96%	$34	168%
Q24	$41	$17	42%	$30	73%
Q09	$22	$27	120%	$47	211%
Q03	$70	$41	58%	$71	102%
Q04	$48	$41	84%	$71	147%
Q26	$118	$55	47%	$97	82%
Q25	$161	$49	30%	$85	53%
Q22	$228	$86	38%	$151	66%
Q21	$269	$47	17%	$82	30%
Q15	$501	$85	17%	$149	30%
Q01	$689	$66	10%	$115	17%
Q02	$443	$52	12%	$91	21%
Q06	$721	$13	2%	$23	3%
Total and Ratios					
	3,331	$597	18%	$1,044	31%
Totals and Ratios excluding. Q06					
	2,610	$584	22%	$1,021	39%
Totals and Ratios excluding. Q06 and Q02					
	2,167	$532	25%	$931	43%
	(1)	(2)	(3)	(4)	(5)

Table 4. Batch Query Workload -- User Charges

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
	------Elapsed Sec.s------						
Query	------DBC------			(1)/	(1)/	-----CPU	Sec.s----
Number	DB2	24 AMP	8 AMP	(2)	(3)	DB2	DBC
12	13	26	79	0.48	0.16	3	65
22	24	14	43	1.66	0.55	8	196
4	25	228	684	0.11	0.04	1	2568
28	33	46	137	0.72	0.24	5	312
30	66	335	1006	0.20	0.07	5	4087
24	89	342	1027	0.26	0.09	12	3974
25	145	214	643	0.68	0.23	63	3144
5	185	255	765	0.72	0.24	27	3095
17	241	81	243	2.97	0.99	15	566
29	259	135	406	1.92	0.64	30	1442
18	283	195	586	1.45	0.48	201	2752
3	296	601	1802	0.49	0.16	49	3105
10	420	245	734	1.72	0.57	85	2835
6	607	312	937	1.94	0.65	221	3296
Subtotal and Ratios (request ~< 10 minutes)							
	2685	3031	9093	0.89	0.30	725	31,437
26	714	477	1432	1.49	0.50	312	6768
11	1298	648	1943	2.00	0.67	647	9118
9	2417	429	1288	5.63	1.88	777	7517
23	2573	1004	3012	2.56	0.85	567	17632
8	2852	536	1607	5.33	1.78	189	1999
16	4168	1039	3116	4.01	1.34	1960	16719
27	7272	699	2096	10.4	3.47	590	10000
21	9345	809	2427	11.6	3.85	966	12655
7	9657	2621	7864	3.68	1.23	2293	40614
15	9901	2543	7629	3.89	1.30	2341	40526
20	19901	3567	10700	5.58	1.86	4425	70447
Subtotal and Ratios (request > 10 minutes)							
	70098	14372	43115	4.88	1.63	15067	233995
Totals and Ratios							
	72783	17402	52207	4.18	1.39	15792	265432
	(1)	(2)	(3)	(4)	(5)	(6)	(7)

Table 5. Interactive Query Workload -- Run Statistics

For purposes of the benchmark the queries were run using an IBM supplied dynamic SQL facility called DSNTEP22 in batch. In the Teradata environment the queries were submitted to a 24 AMP DBC/1012 using a Batch TEQUEL utility (BTEQ) which had similar function to DSNTEP22. All measurements were made in essentially standalone mode running the queries sequentially. Table 5 shows the elapsed time results for the two benchmark environments and an estimation of an 8 AMP DBC/1012 environment. The 8 AMP DBC/1012 environment was included to compare the two environments where an equal number of actuators were available to support the processing. In total the DBC/1012 outperformed DB2 by a factor of 4.88 to 1 in elapsed time. However, the performance improvement was greatly skewed toward long running requests. In requests where the DB2 elapsed time was approximately 10 minutes or less the 24 AMP DBC/1012 was outperformed by DB2 on average by approximately 11%, whereas on the longer running queries the DBC/1012 performance was more than 5 time better than DB2.

From a chargeback perspective, DB2 costs were approximately equal to those from an 8 AMP DBC/1012, but the 24 AMP DBC/1012 would have generated only 65% of the DB2 charge as shown in the On-Demand Charges portion of Table 3. If one was to assume that queries running over an hour in either environment would be scheduled off-shift and take advantage of an off-shift pricing differential (which might be a reasonable assumption in a budget constrained organization employing chargeback) then the 24 AMP DBC/1012 cost are 7% greater than DB2's and the 8 AMP DBC/1012 costs would be 15% greater as shown in Charges with Deferred execution portion of Table 3.

5.3 Transaction Processing Workload.

In order to investigate the transaction processing power of the two systems a set of synthetic kernels were used. These kernels used the various relational operations of insert, update, select, and delete. These were used in combination with varying degrees of set processing against both clustered and non-clustered rows. In addition, attempts were made to determine the cost of commit processing for each of the operations. All of the kernels operated on the COVR table which had 6 indices defined on it. One of these was a unique clustering index and the other 5 were non-unique and non-clustering. All of the kernels accessed the table through the unique index.

The various kernels were submitted as separate batch job steps using the DSNTEP22 program and BTEQ in the DB2 and DBC/1012 environments respectively. No true transaction driver such as CICS or IMS was used to generate the workload for this portion of the benchmark. Great care was taken in the order of execution of the INSERT and UPDATE kernels to insure that the table would retain its clustering and not require refreshing after each sequence.

For this portion of the benchmark the DB2 buffer pools were reduced significantly. It would have been unrepresentative to have used the larger buffer pools without having had significant levels of concurrent transaction processing. Since generating a heavy concurrent transaction workload with facilities such as IMS or CICS would have greatly increased the complexity of transporting the benchmark, the smaller buffer pool strategy was chosen to, in effect, make DB2 churn buffers with single thread transactions. The DB2 parameters were set to provide 20 data buffers in BP0 and 10 index buffers in BP1.

It should also be noted that all requests used the DB2 dynamic SQL facilities which implies that the overhead of developing an execution plan was incurred for every SQL statement processed. In a true transaction processing environment static SQL would undoubtedly be used. This decision was again made to increase the portability of the benchmark to the Teradata benchmark facility.

5.3.1 Delete operations.

Three streams of delete operations were performed. The first deleted increasing numbers of clustered rows from a table. The second run deleted increasing numbers of unclustered rows. The third run was the same as the second except that commits were inserted between each delete operation. The results of the three runs are shown in the sections of Table 6.

In the DB2 environment delete operations require approximately 23 ms ((3.12-.26)/127)of 3090-400 SMF recorded CPU time to delete a clustered row, reaching a throughput of approximately 1.7 rows deleted per second as the size of the set deleted increased. The DBC/1012 managed to deliver set deletes at a maximum throughput of 2.3 rows per second. In the unclustered delete operations the DBC/1012 just slightly outperformed DB2 when commits were deferred to the end of the logical transaction but showed a marked slow down when commits were performed after each row deleted.

Rows Deleted	DB2 CPU sec	DB2 Rows/ CPU sec	DB2 Elapsed sec	DB2 Rows /sec	DBC Elapsed sec	DBC Rows /sec
Clustered rows set deleted						
1	0.26	3.9	0.9	1.07	1.0	1.1
2	0.28	7.1	1.4	1.40	1.2	1.7
4	0.32	12.5	2.6	1.52	2.7	1.5
8	0.41	19.5	4.8	1.67	3.5	2.3
16	0.60	26.7	9.7	1.65	7.0	2.3
32	0.97	33.2	18.7	1.71	15.0	2.1
64	1.70	37.8	36.3	1.76	27.5	2.3
128	3.12	41.1	73.9	1.73	56.5	2.3
Unclustered rows with one commit per row						
1	0.26	3.9	0.9	1.07	1.0	1.1
2	0.32	6.3	1.8	1.13	0.7	2.9
4	0.46	8.7	3.7	1.07	2.3	1.7
8	0.74	10.8	7.6	1.06	6.0	1.3
Unclustered rows with one commit per set of rows						
1	0.26	3.9	0.9	1.07	1.0	1.1
2	0.33	6.1	2.2	0.89	1.3	1.5
4	0.47	8.5	3.7	1.09	4.0	1.0
8	0.77	10.4	7.9	1.01	8.0	1.0

Table 6. Transaction Benchmark - Delete Results

5.3.2 Insert operations.

Four streams of insert operations were performed. The first two performed clustered set inserts on increasing numbers of rows. The second of these included commits after each insert operation. The last two runs were similar to the first two except they operated on rows which were not clustered. Their results are shown in Table 7.

The clustered inserts required 73 ms/row of SMF recorded 3090-400 CPU time with the deferred commits, and 82 ms/row with commits for each insert. By taking the differences in CPU time between the first and second runs the commit overhead can be calculated to about 9 ms. Both DB2 and the DBC/1012 leveled off to approximately 1 row inserted per elapsed second with DB2 being just a little faster. Throughput dropped by a factor of approximately 3 for the DBC/1012 when the inserts became unclustered.

5.3.3 Select Operations

Two runs were made for select processing. The first shows the effect of increasing the number of rows in clustered selects, with the second run performing the same sorts of operations but on unclustered rows. The results are shown in Table 8.

The DB2 system uses about 3 ms/row to select clustered rows and 6 ms/row of SMF recorded 3090-400 CPU time to select unclustered rows. In terms of responsiveness, DB2 far outperformed the DBC/1012 by factor of at least 7 to 1. This was somewhat of a mystery considering that the DBC/1012 compared favourably on most of the other kernels.

5.3.4 Update operation.

Three runs of updates were performed. The first run performed clustered set updates on increasing number of rows. The second run performed updates of increasing numbers of unclustered rows, while the last one added commits between each row update. The results are shown in Table 9.

DB2 uses approximately 8 ms/row of 3090-400 SMF recorded CPU time to update clustered rows with set operations. This can be contrasted with 154 ms/row and 168 ms/row for unclustered updates depending on where the commits are processed. From the response point of view the DBC/1012 and DB2 function approximately the same, with both showing almost a 10 times increase in response time for unclustered updates:

Facilities such as LOCK TABLE were not used before running the update kernels. It was felt that use of these facilities, which can notably improve update performance, would not have been representative of a transaction environment which would have to support a combination of concurrent update and retrieval operations.

Rows Inserted	DB2 CPU sec	DB2 Rows /CPU sec	DB2 Elapsed sec	DB2 Rows /sec	DBC Elapsed sec	DBC Rows /sec
Clustered row inserts -- one commit per set						
1	0.28	3.6	4.1	0.24	3.1	0.33
2	0.34	5.9	1.8	1.13	4.0	0.50
4	0.49	8.2	3.6	1.12	6.0	0.67
8	0.78	10.3	7.0	1.14	9.0	0.89
16	1.38	11.6	14.5	1.10	16.0	1.00
32	2.59	12.4	29.0	1.10	32.0	1.00
64	4.92	13.0	57.7	1.11	66.0	0.97
128	9.57	13.4	115.6	1.11	141.0	0.91
Clustered row inserts -- one commit per row						
1	0.28	3.6	4.1	0.24	3.1	0.33
2	0.34	5.9	1.8	1.09	6.0	0.33
4	0.49	8.2	4.1	0.99	5.0	0.80
8	0.83	9.6	7.3	1.10	9.0	0.89
16	1.48	10.8	14.8	1.08	19.0	0.84
32	2.78	11.5	29.2	1.10	37.0	0.86
64	5.47	11.7	65.6	0.98	69.0	0.93
128	10.71	12.0	120.1	1.07	143.0	0.90
Unclustered row inserts -- one commit per set						
1	0.28	3.6	4.1	0.24	3.1	0.33
2	0.34	5.9	1.8	1.09	4.0	0.50
4	0.49	8.2	3.6	1.13	9.0	0.44
8	0.80	10.0	7.3	1.10	25.0	0.32
Unclustered row inserts -- one commit per row						
1	0.28	3.6	4.1	0.24	3.1	0.33
2	0.35	5.7	1.8	1.11	6.0	0.33
4	0.51	7.8	3.6	1.10	12.0	0.33
8	0.84	9.5	7.5	1.07	23.0	0.35

Table 7. Transaction Benchmark - Insert Results

Rows Selected	DB2 CPU sec	DB2 Rows /CPU sec	DB2 Elapsed sec	DB2 Rows /sec	DBC Elapsed sec	DBC Rows /sec
Clustered selects						
1	0.24	4.2	0.20	4.92	1.5	0.7
2	0.24	8.3	0.15	13.33	5.8	0.3
4	0.25	16.0	0.17	24.24	3.0	1.3
8	0.28	29.1	0.27	30.19	3.8	2.1
16	0.29	55.2	0.28	58.18	6.0	2.7
32	0.34	94.1	0.36	90.14	7.5	4.3
64	0.45	142.2	0.72	88.89	11.5	5.6
128	0.65	198.4	0.89	144.63	21.5	6.0
Unclustered selects						
1	0.24	4.2	0.20	4.92	1.5	0.7
2	0.30	6.7	0.34	5.88	8.0	0.3
4	0.43	9.3	0.80	5.00	8.0	0.5
8	0.67	11.9	1.61	4.97	17.0	0.5

Table 8. Transaction Benchmark - Select Results

5.3.5 Transaction Processing Cost.

In terms of transaction oriented responsiveness the DBC/1012 and DB2 are relatively equal with the exception of SELECT processing. However, in terms of the chargeback cost to accomplish those same operations the DBC/1012 is significantly more attractive. The Table 10 shows the relative costs of performing each of the four basic operations on 255 rows of data. Depending on the size of the configuration the DBC can be about 2 to 4 time more cost effective.

Rows Updated	DB2 CPU sec	DB2 Rows /CPU sec	DB2 Elapsed sec	DB2 Rows /sec	DBC Elapsed sec	DBC Rows /sec
Clustered set updates						
1	0.35	2.9	1.59	0.6	2.3	0.4
2	0.35	5.7	1.82	1.1	1.8	1.1
4	0.37	10.8	2.07	1.9	0.9	4.4
8	0.41	19.5	2.87	2.8	2.7	3.0
16	0.47	34.0	4.39	3.6	5.0	3.2
32	0.61	52.5	7.62	4.2	9.0	3.6
64	0.89	71.9	12.70	5.0	12.0	5.3
128	1.39	92.1	24.56	5.2	26.0	4.9
Unclustered set updates						
1	0.35	2.9	1.59	0.63	2.3	0.4
2	0.51	3.9	3.22	0.62	5.0	0.4
4	0.83	4.8	6.33	0.63	7.0	0.6
8	1.43	5.6	12.41	0.64	11.0	0.7
Unclustered set updates -- with one commit per row						
1	0.35	2.9	1.59	0.63	2.3	0.4
2	0.51	3.9	3.37	0.59	3.0	0.7
4	0.85	4.7	6.67	0.60	4.0	1.0
8	1.53	5.2	13.31	0.60	10.0	0.8

Table 9. Transaction Benchmark - Update Results

Operation	3090 CPU sec	3090 COST	Amdahl 5860 sec	DBC 8 AMP sec	% COST of DB2	24 AMP COST	% of DB2
SET DELETES	7.68	4.61	.77	36	1.37	30	0.88	19
INSERTS	22.38	13.43	16.28	57	4.42	33	3.65	27
SELECTS	2.74	1.64	1.64	29	1.25	76	0.86	52
SET UPDATES	4.84	2.90	.80	25	0.99	34	0.65	22

Table 10. Transaction Benchmark - Charge Comparisons

5.4 Table Loads.

In the DB2 environment the tables were loaded specifying LOG=NO. The release of the DBC software used in the benchmark, provided only a transient transaction log capable of backing out in-flight transactions in the case of hardware or software failure. Table 11 show the results of the load processing in both environments. The CPU time and elapsed time for the creation of all indices associated with a table are included in the numbers for each base table.

DB2 performed better than the DBC/1012, but neither of them were particularly snappy at the job. It took 14 and 18 hours respectively to load about 3 gigabytes of data and indices which is rather difficult to cope with, considering the over-night batch processing windows many shops face. Load times of this duration also make the regular testing of disaster backup procedures very difficult.

Table Name	Table Size in MB	000's Of Rows	DB2 Host CPU Seconds To Load	DB2 Rows/ CPU Second	DB2 Elapsed Seconds To Load	DBC Elapsed Seconds To Load	DB2 Rows/ Elapsed Second	DBC Rows/ Elapsed Second	Elap. Ratio DB2/ TD
BRAN	0.0	0.2	1	402	26	135	8	2	0.19
CMAC	2.4	15.5	12	1335	99	496	157	31	0.20
CAGD	3.2	113.6	140	809	297	569	383	200	0.52
APCO	4.2	173.8	205	849	385	1045	452	166	0.37
IFPA	5.5	155.8	236	660	800	1035	195	151	0.77
INAP	5.5	147.8	280	528	1380	749	107	197	1.84
PLON	13.7	597.8	909	658	6053	3343	99	179	1.81
CIAR	35.1	1464.0	2050	714	3511	7711	417	190	0.46
COIN	39.5	1464.0	2069	708	3464	3288	423	445	1.05
BLAR	70.9	621.5	899	691	1683	3400	369	183	0.49
CLBA	192.1	1288.9	642	2007	3319	7049	388	183	0.47
PLCY	214.2	624.5	1456	429	4478	8234	139	76	0.54
EVEN	230.0	931.2	2977	313	5829	6414	160	145	0.91
CLAD	324.7	1341.5	695	1930	3652	9149	367	147	0.40
COVR	390.5	1654.7	4604	359	15603	11857	106	140	1.32
Totals and Ratios									
	1531.3	10594.8	17176	617	50578	64474	209	164	0.78

Table 11. Comparison of Table Load Performance

5.5 Secondary Index Creation

The parallel processing power of the DBC/1012 showed through in the area of secondary index creation after table load. This is shown in some samples of index creation performance measures in Table 12. The DBC/1012 outperformed DB2 over 4 to 1 in elapsed time.

Base Table Name	Table Size In MB	000's Of Rows	DB2 Host CPU Time (Sec.)	DB2 Rows/ CPU Sec.	DB2 Elapsed Time (Sec.)	DBC Elapsed Time (Sec.)	DB2 Rows/ Elapsed Sec.	DBC Rows/ Elapsed Sec.	Elap. Ratio DB2/ TD
INAP	5.5	147.8	107	1379	135	59	1095	2510	2.29
INAP	5.5	147.8	110	1350	138	54	1067	2723	2.55
EVEN	230.0	931.2	741	1256	1230	280	757	3331	4.40
COVR	390.5	1654.7	1318	1255	1715	645	965	2565	2.66
COVR	390.5	1654.7	1390	1191	1650	517	1003	3200	3.19
PLCY	214.2	624.5	655	953	2424	290	258	2153	8.36
PLCY	214.2	624.5	670	932	3064	495	204	1261	6.19
PLON	13.7	597.8	561	1065	5297	936	113	638	5.66
Totals and Ratios									
	1464.1	6383.0	5553	1150	15653	3277	408	1948	4.78

Table 12. Comparison of Secondary Index Creation Performance after Table Load

6.0 CONCLUSIONS.

When considering a wide variety of workload situations, neither the software nor hardware based relational data-base processing systems appears as a clear performance winner. The two systems benchmarked in this paper each has its own particular strengths. DB2 was shown to outperform the DBC/1012 especially in the area of small decision support query applications and somewhat in the transaction processing area. The DBC/1012 strong suit is in the area of large table scans which are likely to be the realm of complex decision support and analysis systems where it can use its parallel processing muscle.

An 8X24X24 DBC/1012 appears to be approximately equal to the performance of a 3090-200 in processing complex concurrent batch queries at about one quarter the cost. When processing simple interactive queries the 24 AMP DBC/1012 performs comparably to DB2 when no multiprogramming takes place. It is likely, given the degradation patterns shown from the batch query environment, that a much larger DBC/1012 would be necessary to match the capabilities of a 3090-200 for simple interactive query workloads.

7.0 ACKNOWLEDGEMENTS.

The authors would like to thank Pat Gowryluk for his efforts in preparing the data used in the benchmark, his help in explaining the operational environment in which the queries were used, and the actual running of the queries during the benchmark. As well, much gratitude is owed Craig Essex, for without whom, none of the preceding data would have been available for analysis.

REFERENCES

[1] IBM, Database 2 General Information Manual, GC26-4073-2 (1986).

[2] Teradata Corporation, DBC/1012 Database Computer Concepts and Facilities, C02-0001 (1986).

[3] Teradata Corporation, DBC/1012 Database Computer User's Guide, C09-0001 (1986).

[4] Teradata Corporation, DBC/1012 Database Computer Systems Manual, C010-0001 (1986).

APPENDIX A

Query #	Tables Joined	SQL Functions Used
3	3	COUNT, SUM
4	7	ORDER, + and - operators
5	7	ORDER
6	8	SUM, GROUP, ORDER, table joined to itself
7	6	SUM, GROUP, ORDER
8	1	SUM, UNION, GROUP, SUBSELECT, DISTINCT
9	1	COUNT, GROUP, ORDER, HAVING
10	4	GROUP, ORDER
11	2	BETWEEN
12	1	IN
14	1	UPDATE
15	6	SUM, GROUP, ORDER
16	3	ORDER
17	2	ORDER
18	2	COUNT, DISTINCT, GROUP, ORDER
20	3	GROUP, ORDER
21	5	ORDER
22	1	GROUP, UNION, AVG, COUNT, MAX, MIN, union of a table to itself
23	1	COUNT, DISTINCT
24	8	ORDER
25	6	DISTINCT, ORDER
26	3	GROUP
27	6	ORDER
28	7	ORDER
29	7	ORDER, DISTINCT
30	8	ORDER

List of SQL Functions used in Interactive Query Benchmark

EXPLAIN Output Listings for the Interactive Query Benchmark

QUERY NO	Q BLCK NO	PLAN NO	METHOD	TNAME	TAB NO	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	SORTN UNIQ	SORTN JOIN	SORTN ORDER BY	SORTN GROUP BY	SORTC UNIQ	SORTC JOIN	SORTC ORDER BY	SORTC GROUP BY	TS LOCK MODE
1	1	1	0	COVR	1	I	0	COVRI6	Y	N	N	N	N	N	N	N	N	IS
																		*
2	1	1	0	COVR	1	I	0	COVRI7	Y	N	N	N	N	N	N	N	N	IS
																		*
3	1	1	0	EVEN	3	R	0		N	N	N	N	N	N	N	N	N	IS
	1	2	1	PLCY	1	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	3	1	COVR	2	I	1	COVRI1	N	N	N	N	N	N	N	N	N	IS
	1	4	3		0		0		N	N	N	N	N	N	N	N	N	
																		*
4	1	1	0	COVR	3	I	1	COVRI4	N	N	N	N	N	N	N	N	N	IS
	1	2	1	PLCY	1	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	3	1	INAP	2	I	1	INAPI5	N	N	N	N	N	N	N	N	N	IS
	1	4	1	COIN	6	I	2	COINI1	Y	N	N	N	N	N	N	N	N	IS
	1	5	1	CLBA	4	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	6	1	BRAN	5	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	7	1	APCO	7	I	3	APCOI1	Y	N	N	N	N	N	N	N	N	IS
	1	8	3		0		0		N	N	N	Y	N	N	N	N	N	
																		*
5	1	1	0	COVR	2	I	1	COVRI6	N	N	N	N	N	N	N	N	N	IS
	1	2	1	PLCY	1	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	3	1	CAGD	5	I	1	CAGDI1	Y	N	N	N	N	N	N	N	N	IS
	1	4	1	CMAC	7	I	1	CMACI1	N	N	N	N	N	N	N	N	N	IS
	1	5	1	PLON	6	I	1	PLONI1	Y	N	N	N	N	N	N	N	N	IS
	1	6	1	CLBA	4	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	7	1	INAP	3	I	1	INAPI5	N	N	N	N	N	N	N	N	N	IS
	1	8	3		0		0		N	N	N	Y	N	N	N	N	N	
																		*
6	1	1	0	COVR	3	I	1	COVRI4	N	N	N	N	N	N	N	N	N	IS
	1	2	1	PLCY	1	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	3	1	COVR	2	I	1	COVRI1	N	N	N	N	N	N	N	N	N	IS
	1	4	1	BRAN	4	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	5	1	CAGD	8	I	1	CAGDI1	Y	N	N	N	N	N	N	N	N	IS
	1	6	1	CMAC	7	I	1	CMACI1	N	N	N	N	N	N	N	N	N	IS
	1	7	1	PLON	5	I	1	PLONI1	Y	N	N	N	N	N	N	N	N	IS
	1	8	1	CLBA	6	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	9	3		0		0		N	N	N	N	N	N	N	N	Y	
																		*
7	1	1	0	COVR	2	I	1	COVRI4	N	N	N	N	N	N	N	N	N	IS

87/02/13 10:33 PAGE 1

QUERY NO	Q BLCK NO	PLAN NO	METHOD	TNAME	TAB NO	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	SORTN UNIQ	SORTN JOIN	SORTN ORDER BY	SORTN GROUP BY	SORTC UNIQ	SORTC JOIN	SORTC ORDER BY	SORTC GROUP BY	TS LOCK MODE
7	1	2	1	PLCY	3	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	3	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	4	1	CMAC	6	I	1	CMACI1	N	N	N	N	N	N	N	N	N	IS
	1	5	1	PLON	4	I	1	PLONI1	Y	N	N	N	N	N	N	N	N	IS
	1	6	1	CLBA	5	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	7	3		0		0		N	N	N	N	N	N	N	N	Y	
																		*
8	2	1	0	COVR	2	I	1	COVRI4	N	Y	N	N	N	N	N	N	N	IS
	3	1	0	COVR	1	I	1	COVRI4	N	N	N	N	Y	N	N	N	N	IS
	1	1	0	COVR	1	I	1	COVRI1	N	N	N	N	Y	N	N	N	N	IS
	3	2	3		0		0		N	N	N	N	N	Y	N	N	N	
																		*
9	1	1	0	EVEN	1	R	0		N	N	N	N	Y	N	N	N	N	IS
																		*
10	1	1	0	COVR	3	I	1	COVRI4	N	N	N	N	N	N	N	N	N	IS
	1	2	1	PLCY	2	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	3	1	INAP	4	I	1	INAPI5	N	N	N	N	N	N	N	N	N	IS
	1	4	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	5	3		0		0		N	N	N	N	N	N	N	N	Y	
																		*
11	1	1	0	PLCY	2	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	2	1	COVR	1	I	1	COVRI1	N	N	N	N	N	N	N	N	N	IS
																		*
12	1	1	0	COVR	1	I	1	COVRI1	N	N	N	N	N	N	N	N	N	IS
																		*
13	1	1	0	PLCY	1	R	0		N	N	N	N	N	N	N	N	N	S
	2	1	0	COVR	2	I	1	COVRI4	N	Y	N	N	N	N	N	N	N	IS
																		*
15	1	1	0	COVR	2	I	1	COVRI4	N	N	N	N	N	N	N	N	N	IS
	1	2	1	PLCY	3	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	3	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	4	1	CMAC	6	I	1	CMACI1	N	N	N	N	N	N	N	N	N	IS
	1	5	1	PLON	4	I	1	PLONI1	Y	N	N	N	N	N	N	N	N	IS
	1	6	1	CLBA	5	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	7	3		0		0		N	N	N	N	N	N	N	N	Y	

QUERY NO	Q BLCK NO	PLAN NO	METHOD	TNAME	TAB NO	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	SORTN UNIQ	SORTN JOIN	SORTN ORDER BY	SORTN GROUP BY	SORTC UNIQ	SORTC JOIN	SORTC ORDER BY	SORTC GROUP BY	TS LOCK MODE
16	1	1	0	CLBA	3	I	0	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	2	1	PLON	2	I	1	PLONI2	N	N	N	N	N	N	N	N	N	IS
	1	3	1	PLCY	1	I	1	PLCYI1	Y	N	N	N	N	N	N	N	N	S
																		*
17	1	1	0	PLCY	1	R	0		N	N	N	N	N	N	N	N	N	S
	1	2	1	BRAN	2	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	3	3		0		0		N	N	N	Y	N	N	N	N	N	
																		*
18	1	1	0	EVEN	1	R	0		N	N	N	N	N	N	N	N	N	IS
	1	2	1	BRAN	2	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	3	3		0		0		N	N	N	N	N	N	N	N	Y	
																		*
19	1	1	0	PLON	1	I	0	PLONI1	N	N	N	N	N	N	N	N	N	IS
	1	2	1	CLAD	4	I	2	CLADI1	N	N	N	N	N	N	N	N	N	IS
	1	3	1	CLBA	3	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	4	1	BLAR	2	I	1	BLARI2	N	N	N	N	N	N	N	N	N	IS
	1	5	3		0		0		N	N	N	Y	N	N	N	N	N	
																		*
20	1	1	0	COIN	2	R	0		N	N	N	N	N	N	N	N	N	IS
	1	2	1	CIAR	3	I	3	CIARI1	Y	N	N	N	N	N	N	N	N	IS
	1	3	1	CLAD	1	I	2	CLADI1	N	N	N	N	N	N	N	N	N	IS
	1	4	3		0		0		N	N	N	N	N	N	N	N	Y	
																		*
21	1	1	0	PLCY	2	I	1	PLCYI6	N	N	N	N	N	N	N	N	N	S
	1	2	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	3	1	PLON	4	I	1	PLONI1	N	N	N	N	N	N	N	N	N	IS
	1	4	1	CLAD	5	I	2	CLADI1	N	N	N	N	N	N	N	N	N	IS
	1	5	1	CLBA	3	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	6	3		0		0		N	N	N	Y	N	N	N	N	N	
																		*
22	1	1	0	CMAC	1	R	0		N	N	N	N	Y	N	N	N	N	IS
	2	1	0	CMAC	1	R	0		N	N	N	N	Y	N	N	N	N	IS
	3	1	0	CMAC	1	R	0		N	N	N	N	Y	N	N	N	N	IS
	3	2	3		0		0		N	N	N	N	N	Y	N	N	N	
	2	2	3		0		0		N	N	N	N	N	Y	N	N	N	

QUERY NO	Q BLCK NO	PLAN NO	METHOD	TNAME	TAB NO	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	SORTN UNIQ	SORTN JOIN	SORTN ORDER BY	SORTN GROUP BY	SORTC UNIQ	SORTC JOIN	SORTC ORDER BY	SORTC GROUP BY	TS LOCK MODE
23	1	1	0	COVR	1	R	0		N	N	N	N	N	N	N	N	N	IS
24	1	1	0	COVR	3	I	1	COVRI4	N	N	N	N	N	N	N	N	N	IS
	1	2	1	CIAR	4	I	2	CIARI1	Y	N	N	N	N	N	N	N	N	IS
	1	3	1	PLCY	2	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	4	1	CAGD	8	I	1	CAGDI1	Y	N	N	N	N	N	N	N	N	IS
	1	5	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	6	1	CMAC	5	I	1	CMACI1	N	N	N	N	N	N	N	N	N	IS
	1	7	1	CLBA	6	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	8	1	CCTT	7	I	1	CCTTI1	N	N	N	N	N	N	N	N	N	IS
	1	9	3		0		0		N	N	N	Y	N	N	N	N	N	
25	1	1	0	INAP	5	R	0		N	N	N	N	N	N	N	N	N	IS
	1	2	1	COVR	3	I	1	COVRI1	N	N	N	N	N	N	N	N	N	IS
	1	3	1	PLCY	2	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	4	1	CIAR	4	I	2	CIARI1	Y	N	N	N	N	N	N	N	N	IS
	1	5	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	6	1	CLBA	6	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	7	3		0		0		N	Y	N	Y	N	N	N	N	N	
26	1	1	0	COVR	2	I	1	COVRI4	N	N	N	N	N	N	N	N	N	IS
	1	2	1	INAP	3	I	1	INAPI5	N	N	N	N	N	N	N	N	N	IS
	1	3	1	PLCY	1	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	4	3		0		0		N	N	N	N	N	N	N	N	Y	
27	1	1	0	PLCY	2	I	1	PLCYI2	N	N	N	N	N	N	N	N	N	S
	1	2	1	COVR	3	I	1	COVRI1	N	N	N	N	N	N	N	N	N	IS
	1	3	1	INAP	5	I	1	INAPI5	N	N	N	N	N	N	N	N	N	IS
	1	4	1	APCO	6	I	3	APCOI1	Y	N	N	N	N	N	N	N	N	IS
	1	5	1	EVEN	4	I	1	EVENI5	N	N	N	N	N	N	N	N	N	IS
	1	6	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	7	3		0		0		N	N	N	Y	N	N	N	N	N	
28	1	1	0	INAP	3	R	0		N	N	N	N	N	N	N	N	N	IS
	1	2	1	COVR	2	I	1	COVRI1	N	N	N	N	N	N	N	N	N	IS
	1	3	1	PLCY	6	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	S
	1	4	1	CIAR	5	I	2	CIARI1	Y	N	N	N	N	N	N	N	N	IS
	1	5	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS

QUERY NO	Q BLCK NO	PLAN NO	METHOD	TNAME	TAB NO	ACCESS TYPE	MATCH COLS	ACCESS NAME	INDEX ONLY	SORTN UNIQ	SORTN JOIN	SORTN ORDER BY	SORTN GROUP BY	SORTC UNIQ	SORTC JOIN	SORTC ORDER BY	SORTC GROUP BY	TS LOCK MODE
28	1	6	1	CLBA	4	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	7	1	CCTT	7	I	1	CCTTI1	N	N	N	N	N	N	N	N	N	IS
	1	8	3		0		0		N	N	N	Y	N	N	N	N	N	
29	1	1	0	PLCY	2	R	0		N	N	N	N	N	N	N	N	N	S
	1	2	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	3	1	EVEN	5	I	1	EVENI5	N	N	N	N	N	N	N	N	N	IS
	1	4	1	CAGD	6	I	2	CAGDI1	Y	N	N	N	N	N	N	N	N	IS
	1	5	1	CMAC	7	I	1	CMACI1	N	N	N	N	N	N	N	N	N	IS
	1	6	1	CIAR	4	I	2	CIARI1	Y	N	N	N	N	N	N	N	N	IS
	1	7	1	CLBA	3	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	8	3		0		0		N	Y	N	Y	N	N	N	N	N	
30	1	1	0	COVR	4	I	1	COVRI4	N	N	N	N	N	N	N	N	N	IS
	1	2	1	PLCY	2	I	1	PLCYI1	N	N	N	N	N	N	N	N	N	IS
	1	3	1	APCO	6	I	1	APCOI2	N	N	N	N	N	N	N	N	N	IS
	1	4	1	BRAN	1	I	1	BRANI1	N	N	N	N	N	N	N	N	N	IS
	1	5	1	CAGD	5	I	1	CAGDI1	Y	N	N	N	N	N	N	N	N	IS
	1	6	1	CMAC	8	I	1	CMACI1	N	N	N	N	N	N	N	N	N	IS
	1	7	1	PLON	7	I	1	PLONI1	Y	N	N	N	N	N	N	N	N	IS
	1	8	1	CLBA	3	I	1	CLBAI1	N	N	N	N	N	N	N	N	N	IS
	1	9	3		0		0		N	N	N	Y	N	N	N	N	N	

PERFORMANCE TUNING FOR A VAX-BASED RELATIONAL DATABASE APPLICATION: A CASE STUDY AND LESSONS LEARNED

G. Burgess Allison
The MITRE Corporation

ABSTRACT

This paper describes the steps taken to optimize performance of a relational database application. The application is intended for correspondence management; it is strongly retrieve-and-display oriented; and it was initially designed to support interactive access to approximately 50,000 logical records by a limited number of users. The processing environment is a VAX cluster, with Relational Technology, Inc.'s INGRES as the application DBMS. The paper includes a description of the performance-related problems encountered, the system design approaches that were adopted during application development, and the implementation-level approaches that were incorporated while the system was in use. Performance was significantly improved without modifications to either the hardware or DBMS: queries against an initial 53-record test data base failed to complete in 23 hours; the average append and retrieve response times on the production system was lowered to 11 seconds. The paper concludes by identifying several key factors that can dramatically affect performance in a relational database application.

1. INTRODUCTION

The relational data base model is currently enjoying widespread popularity. Indeed, there are several compelling advantages to the relational model; hardware and software vendors are emphasizing relational capabilities; and performance improvements in relational products recently prompted the acknowledged originator of the relational model, E.F. Codd, to conclude, "nonrelational DBMSes have finally become obsolete!"[1]

Regardless of arguments for and against the relational model, many application developers and prototypers are finding themselves in situations where they *must* use a relational DBMS; and there are some applications that produce dramatically poor performance when implemented in a relational environment.

This paper presents a case study of just such an application—initial test queries failed to complete in 23 hours—together with the techniques that were used to improve performance to acceptable levels. Some of the techniques used violate basic principles of the relational model; some are traditional performance tuning techniques; and a few are specific to the DBMS used. In the production system eventually implemented, the average response time was reduced to 11 seconds.

Some of the experience from this performance tuning process was undoubtedly unique to the application. However, some of the techniques and results may have more general applicability and may be of use to developers and designers who are faced with the task of implementing similar applications in a relational environment, or who have an opportunity to select the data base model to be used. It will be the purpose of this paper to focus on the experience with such general applicability.

1.1 BACKGROUND

The application discussed in this paper was developed to provide automated assistance in the process of correspondence management for a large Federal agency. This process includes: review and distribution of all incoming mail; assignment of responsibility for answering significant correspondence; tracking the progress of such assignments; and coordinating the generation, review, and signature of outgoing correspondence. The automated components of this process include: storage of information contained in the correspondence; tracking of assigned responses and outgoing correspondence; and retrieval of archives data in response to ad hoc inquiries.

The correspondence and files unit (CFU) of this particular agency[2] was already using an automated system to assist in correspondence management activities. The existing system was proprietary, and was maintained on a dedicated minicomputer. A new automated correspondence management system (ACMS) was intended to replace and upgrade the proprietary system—as an application on the agency's Digital Equipment Corp. (DEC) VAX cluster, using Relational Technology Inc.'s INGRES relational data base management system (DBMS). Thus, the development project included two phases: pre-implementation and post-implementation. The ACMS is currently operational in a production environment.

1.2 PURPOSE

This paper documents the steps that were taken to optimize performance of the ACMS. The paper describes the performance-related problems that were encountered, the system design approaches that were adopted during application development to optimize performance, and the implementation-level approaches that were incorporated while the system was in use. The rationale for each approach and the performance tests conducted for each are also described.

Since this is a case study, it is not the purpose of this paper to define an ideal performance optimization process; rather the purpose is to record what was actually done. Indeed, there were instances during the project when limited resources precluded extensive pre-testing; when decisions were made because a conclusion was "intuitively obvious"; and when time constraints—caused by trying to conduct performance tuning on an operational system—dictated short-cuts in the construction of tests and in the control over test conditions. Still, in many cases the findings are clear, and alternative approaches that were identified during post-project review are presented in the last section as lessons learned.

1.3 SCOPE

The scope of performance optimization in this project was expressly limited to application-level programming, using only documented INGRES features in a VAX cluster environment. Also, the ACMS was implemented as one of several applications running on the VAX cluster;

---

[1] Codd, E.F. "1987: The Beginning Of A New Era In Data Processing," InformationWEEK, CMP Publications, Inc., Manhasset, NY, May 4, 1987.

[2] The identity of the Federal agency involved is immaterial to the issues under discussion; in the absence of a compelling reason, the identity will not be disclosed.

there was no provision or intention to provide the ACMS application with any "special" status. Consequently, performance tuning in this project was not to include any of the following: "tuning" the VAX to maximize performance of INGRES applications; use of undocumented INGRES features; or modifications to the VAX hardware, the VMS operating system, or INGRES.

Accordingly, the scope of this paper is limited to decisions and approaches at the system design and implementation levels.

1.4 ORGANIZATION OF PAPER

The organization of this paper is as follows:

- Section 2, "SYSTEM DESCRIPTION," describes the environment within which the ACMS operates. This includes hardware and software in the application development environment, and application-level constraints shaped by users and user requirements.
- Section 3, "SYSTEM DESIGN FACTORS," describes performance-related problems that were encountered prior to implementation, the system design factors and alternatives that were considered for resolving each performance problem, and the specific system design approach that was used to resolve each problem.
- Section 4, "IMPLEMENTATION-LEVEL FACTORS," describes the performance problems that were reported in the operational ACMS, the implementation-level factors that were determined to have a significant impact on performance, and the specific techniques and modifications that combined to resolve the performance problems.
- Section 5, "SUMMARY," describes findings from the ACMS performance tuning efforts, and describes general conclusions that may be drawn from these findings.

2. SYSTEM DESCRIPTION

This section describes the environment within which the ACMS operates. This includes hardware and software in the application development environment[3], and application-level constraints shaped by users and user requirements.

2.1 APPLICATION DEVELOPMENT ENVIRONMENT

The ACMS application development and operations environment is illustrated in Figure 1. As shown, the ACMS operates on a pair of DEC VAX minicomputers, connected through DEC's DECNET VAX-cluster hardware and software. The operating system is DEC's VMS. The standard user interface adopted on the VAX cluster is DEC's ALL-IN-1 Office Automation system, and all applications are invoked from an ALL-IN-1 menu.

The ACMS is an INGRES Application-By-Form (ABF) application, using FORTRAN-based Embedded Query Language (EQUEL) programs as ABF procedures and Query-By-Form (QBF) definitions as ABF frames. All forms are defined using INGRES' Visual Forms Editor (VIFRED). EQUEL is a preprocessor which produces VAX VMS FORTRAN source code with appropriate calls to the INGRES DBMS library functions. FORTRAN programs are written by the application developer with specially formatted INGRES statements, and are submitted to the EQUEL preprocessor as input. QBF is a development tool which generates components of an application (such as append, update, and delete) based on a definition of components (e.g., VIFRED form name and INGRES table name) supplied by the application developer. ABF is a menu-driven environment that coordinates user access to EQUEL and QBF modules.

There are several connectivity options that define the ACMS user environment: direct connection to the VAX-cluster DEVELCON switch, which permits the user to log on to either of the two VAX minicomputers; asynchronous connection to the DEVELCON switch, through either dial-up modems or the agency's wide-area network; and asynchronous connection to the wide-area network, through dial-up modems. In all cases, the user must respond to the DEVELCON switch prompts to select a host and must complete the necessary VMS log-in procedures on the selected host. Access is provided to the user at a start-up ALL-IN-1 menu and the ACMS is invoked by a command from the ALL-IN-1 command line.

Direct connection to an individual VAX minicomputer is possible, but is strongly discouraged by VAX systems personnel since it precludes logging onto the "other" VAX and thereby eliminates many of the advantages of the cluster environment.

User workstations include VT220 terminals and DecMate II communicating word processors.

Specific capacities of the hardware, and specific release and version numbers of the software are not shown because it was an express objective that the ACMS would be independent of such factors. Indeed, through the course of the project, VAX memory and disk storage units were changed, the operating system was upgraded to a new major release, and ALL-IN-1 was upgraded to a new major release. Major new releases of INGRES were also installed. These changes, and their effects, will be discussed in appropriate sections of this paper.

2.2 APPLICATION REQUIREMENTS

The correspondence and files unit (CFU) processes more than 40,000 pieces of mail each year. Each piece of correspondence is reviewed by an analyst. All items that require action are recorded and assigned for response. Also, some of the correspondence is judged by the analyst to be "significant," irrespective of a need for action or response, and is recorded for archival and reference purposes. The CFU is responsible for assigning and tracking the progress of responses, for coordinating all outgoing correspondence, and for responding to questions regarding past or pending correspondence.

The ACMS was developed to support three specific functional requirements:

- Record Keeping—storage and update of correspondence data (abstracts and relevant tracking information)
- Tracking—management reports to assist in the tracking of responses
- Reference—selective retrieval of individual correspondence records

A single data form is sufficient to accept correspondence input, to update records, to specify search criteria, and to review retrieved records. For these entry and reference purposes, the full correspondence record is required.

---

[3] The descriptions of systems and products currently in use do not represent endorsements of the product lines or vendors mentioned. We are in no way suggesting them as recommended configurations, nor are we commenting on their relative strength as compared with other possible configurations—available from other vendors, or from the same vendor.

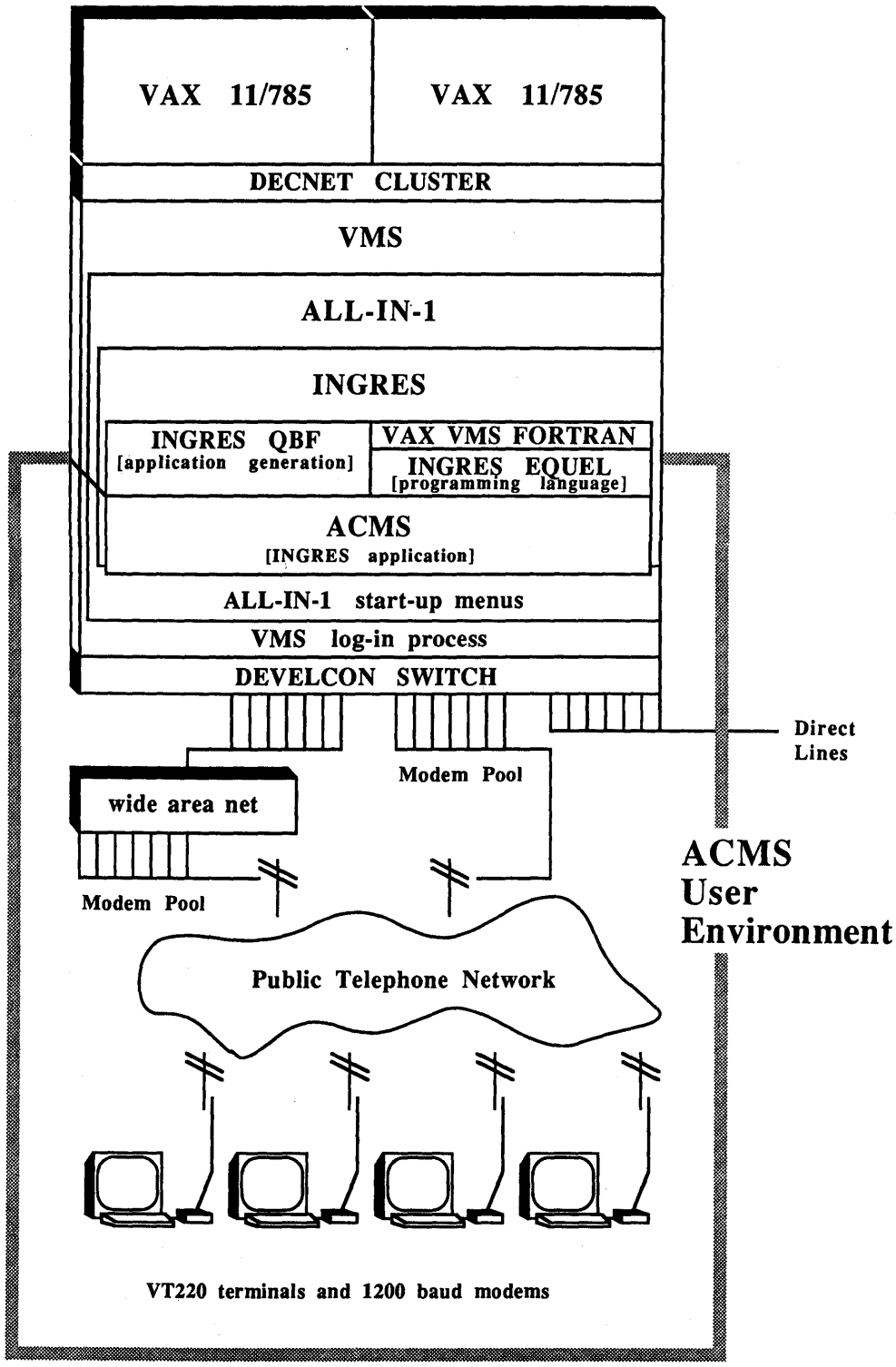


Figure 1. ACMS Application Development Environment

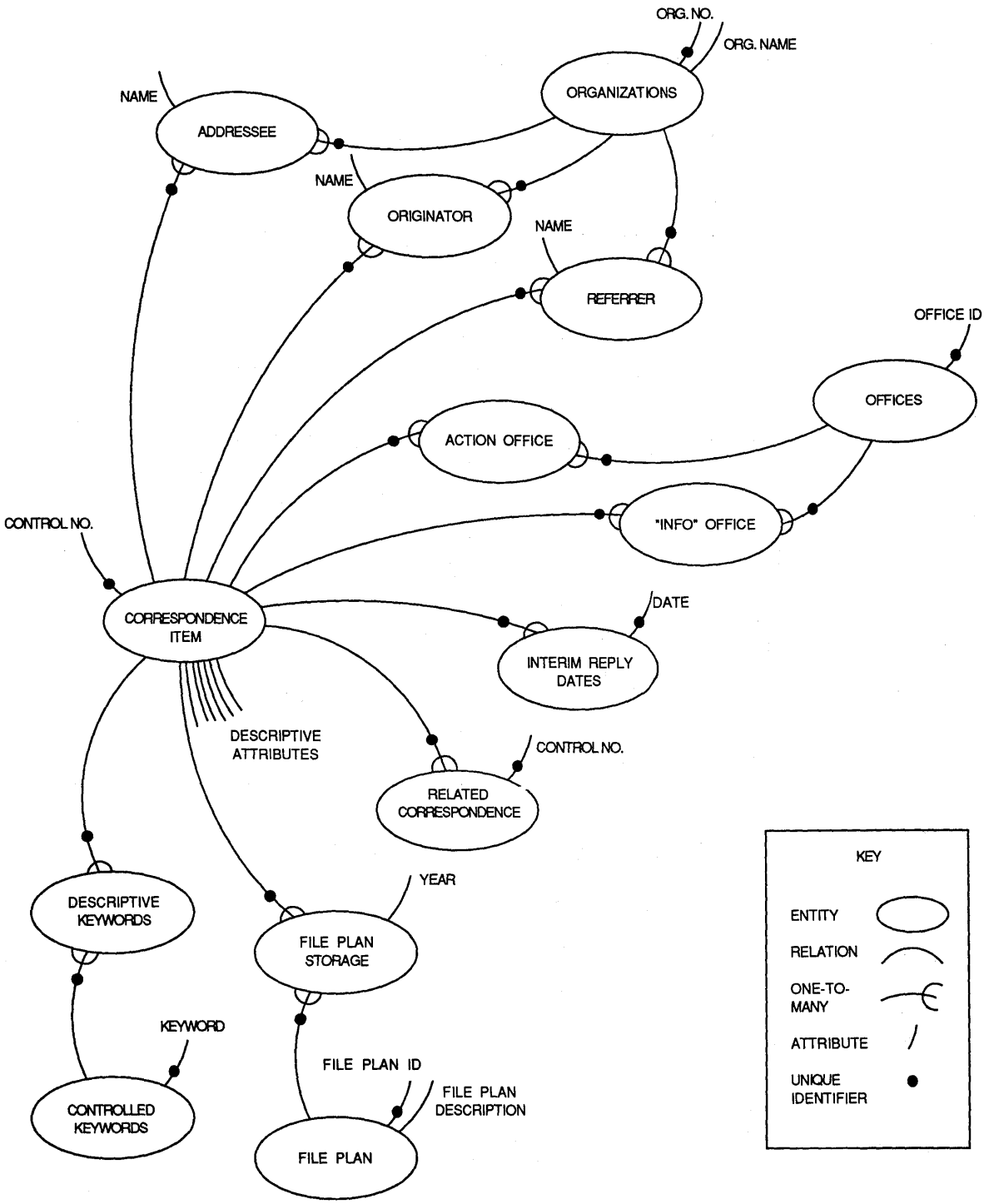


Figure 2. ACMS Data Structure Diagram

Due to the nature of inquiries for information on specific items and classes of correspondence, the retrieval function must be flexible enough to identify qualifying correspondence from partial information (e.g., author, date, subject), and it must be responsive enough to be considered interactive. Specific system-related and performance requirements were identified as follows:

- Archival volume: 50,000 records through 1985.
- Transaction volume: 6,000 to 10,000 records per year.
- Record structure: includes "repeating fields" or "multiples"—a record might show several authors, several keyword descriptors, or several "interim" replies. Analysis of user requirements indicated that there were nine such repeating fields (see below).
- Record size: approximately 1750 characters, depending on occurrence of repeating fields.
- Response time: specified in requirements documentation as, "an interactive process: movement among different modules within the application must be accomplished in a few seconds; pre-defined system operations must be accomplished within 10 to 20 seconds."
- Data entry: some fields require character validation; some fields require table lookup validation; and one field—the correspondence control number, which was defined as the data base primary key—must be unique and automatically assigned by the application.

In retrospect, two of these requirements proved to have a pivotal impact on the performance of the application in a relational environment: the record structure, and the users' need for access to logical records.

For each logical record, there are nine fields and groups of fields which can have multiple entries:

- addressee—the correspondence may be addressed to more than one person. This information includes each addressee's name, and a unique code number for the addressee's organization.
- originator—the correspondence may have been sent from (or signed by) more than one person. This information includes each originator's name and organization code number.
- referrer—the correspondence may have been referred from (or through) more than one person. This information includes each referrer's name and organization code number.
- keyword—the subject matter of the correspondence may be described with several descriptive keywords, taken from a controlled vocabulary.
- action office—responsibility for responding to an item of correspondence may be assigned by the CFU to more than one office within the agency.
- information office—the CFU may forward "information only" copies to several offices within the agency.
- interim reply dates—the CFU or action office(s) may send several interim replies to the correspondent, providing information about the progress of a response.
- related records—the item of correspondence may be related, by subject matter, to other items in the ACMS. The identity of related correspondence is recorded using the CFU control number (primary key).
- "file plan number"—the physical copy of the correspondence is stored in physical files on the basis of a content-sensitive filing system (known as a "file plan") similar in function to the Dewey Decimal System used in libraries. A single item of correspondence may be copied and stored under several file plan numbers.

Strictly adhering to the relational model, the resulting data structure includes 14 separate relational schemas, as illustrated in the data structure diagram in Figure 2. (The diagram does not show fields—or "attributes"—that are simply descriptive in nature.)

A second key requirement was the users' need for access to *all* the information related to a particular item of correspondence, whenever that item was accessed for update or reference. Simply put, the users expressed a strong and certain need for access to the entire logical record: the nature of the inquiries handled by the CFU demanded retrieval of all relevant data, and dividing the retrieval process into several "partial" steps would interfere with the office's work to an unacceptable extent.

How these factors affected performance in a relational environment will be discussed in the next section.

3. SYSTEM DESIGN FACTORS

This section describes performance-related problems that were encountered prior to implementation, the system design factors and alternatives that were considered for resolving each performance problem, and the specific system design approach that was used to resolve each problem.

The principal problem areas encountered include: implementation of the ACMS record structure in a relational data base environment; the amount of ACMS data; the choice of an INGRES application development tool; and the choice of an INGRES data base structure.

All of the pre-implementation performance problems encountered were resolved prior to implementation, and the response times achieved in the development account were within acceptable ranges. Tests conducted in user accounts—through the ALL-IN-1 interface—also produced acceptable results. Terminals were installed in user offices, training was conducted, conversion of data from the existing correspondence management system was completed and the ACMS was implemented as an operational system. Performance problems encountered and resolved after implementation are discussed in Section 4.

3.1 RELATIONAL STRUCTURE OF LOGICAL RECORDS

The relational structure of the ACMS logical records represented the single largest factor having a negative impact on performance. Queries against an initial 53-record test data base would not complete in 23 hours, and would occasionally cause the VAX process to fail—requiring restoration of the entire data base.

The problem was attributable to the implementation of the ACMS "repeating field" record structure in a relational data base. While it is commonly held that the relational model can be used to emulate many different types of data base structures, the use of several related tables to implement repeating fields had a negative and dramatic impact on performance.

The approach used to resolve the problem was to implement most repeating fields by limiting the number of

possible repetitions and defining a separate field for each such repetition. Although the first-normal-form constraint[4] of the relational data base model is not technically violated using this technique, the underlying principles are unequivocally defeated. Under the circumstances, however, the approach was deemed necessary.

3.1.1 Problem Conditions

The initial test data base used 8 related tables[5] and a single primary key[6] that uniquely identified logical records. One of the tables was treated as a "master" table; it contained non-repeating data that pertained to a single item of correspondence (e.g., abstract, date written, date received). The primary key to the master table was defined as unique (i.e., no duplicate keys). The 7 other tables represented "detail" tables; they contained data pertaining to a single item of correspondence for which there might be multiple entries. The primary key for all detail records was the same as that for the related master record, thereby defining a logical ACMS record. Since there would be multiple detail records, duplication of primary keys was permitted in the detail tables. The master table contained 45 fields representing approximately 1500 characters; the 7 detail tables contained 1 to 3 fields each, ranging in length from 1 to 60 characters.

An initial test data base was constructed with 53 logical records—each having a single master record, and most having at least 2 detail records in each of the 7 detail tables. The number of detail records ranged from 1 per detail table to 7 per detail table.

A simple query was constructed to join the eight components of the logical ACMS records (using the common primary key) and to retrieve individual records for review. Response times varied dramatically, depending on the number and distribution of detail records. Logical records with the fewest number of detail records were retrieved in a few seconds. Logical records with high numbers of detail records (e.g., 7 records in each of 7 detail tables) could not be retrieved. Queries for some records would continue until the process was terminated for daily backup procedures (all user-processes on the VAX cluster are terminated each night for backups—the longest attempted query was allowed to run for a total of 23 hours.) Queries for some records would cause INGRES to fail completely and the data base would have to be restored.

Analysis of the relational record structure demonstrates why this occurs.

When a relational join is used to retrieve a logical record from related tables, the data base retrieves a set of de-normalized data (multiple records). For example, if a master record is joined with a related "keyword" table that has 2 qualifying records, the join produces 2 separate records: one showing all the data from the master record plus one of the keywords, and another showing all the master data with the other keyword.

domain of Join =	(master	keyword)
	m_1	k_1
	m_1	k_2

If the master and keyword tables are joined with a related "referral" table that has 3 qualifying records, then the join repeats the master/keyword pair 3 times—once for each referral—producing a total of 6 records.

domain of Join =	(master	keyword	referral)
	m_1	k_1	r_1
	m_1	k_1	r_2
	m_1	k_1	r_3
	m_1	k_2	r_1
	m_1	k_2	r_2
	m_1	k_2	r_3

The total number of records that will be retrieved from a relational join is the product of the several record counts for each of "n" detail tables. Where D_n is the record count for detail table n, the total number of records retrieved in a join for a single logical record is:

$$\text{"join" records} = (D_1) \times (D_2) \times (D_3) \times \ldots \times (D_n)$$

With the structure and data of the test data base—8 related tables with as many as 7 records in each of 7 detail tables—the total number of de-normalized records retrieved for the largest logical record was $(7)^7$, or 823,543 total records. With approximately 1750 characters per logical record, the total amount of data that had to be handled to retrieve that one logical record was 1.44 gigabytes.

Two factors combine to produce the observed performance problem: the number of related tables has a severe impact on the amount of data that must be handled during even simple retrieves; and the relational join is responsible for the multiplicative effect of related tables.

3.1.2 Reduce the Number of Related Tables

The first part of the solution was to reduce the number of related tables. The multiplicative effect of even a small number of such tables (combined with an INGRES limitation[7]) strongly suggested that the number of related tables be reduced to one. Also, the nature of the ACMS usage indicated that one particular field (keywords) would have much greater utility as a "repeating" field than any other data element. Based on these considerations the number of related tables was reduced to one: keywords.

Two different techniques were used to eliminate the remaining repeating fields, depending primarily on anticipated data element usage.

For data elements that are part of repeating groups of fields, or where a structured data entry format is highly valued (such as fields that are likely to be used as search criteria), repeating fields were implemented by selecting a limit to the number of repetitions that would be permitted, then defining that many iterations of the field in the master table. For example, it was determined that three file plan numbers would offer a sufficient number of repetitions and the master record data dictionary was modified to include three separate file plan fields: file_plan1, file_plan2, and file_plan3. This approach permits structuring and validation of entered data, but it also means that a user searching for occurrences of a particular file plan number, for example, would have to repeat the search

---

[4] Each record, or "row," may have only one value for any given field, or "attribute."

[5] The initial test data base was constructed without use of separate tables for "interim reply dates" or "related records."

[6] The correspondence control number was the primary key.

[7] INGRES limits the numbers of related tables that can be displayed on-screen at one time: only one related detail table, referred to in INGRES as a table field, can be displayed on-screen with corresponding master records.

three times: specifying the qualifying criteria in each of the three repeated file plan fields.

For data elements that are likely to be used most for reference (rather than retrieval) purposes, where structured data entry format is less important, repeating fields were implemented by defining a single large field—sufficiently large to permit inclusion of several entries. This single-field approach increases the flexibility for storing multiple entries, but is not conducive to data structuring or validation. Searching such a field is accomplished primarily through the use of bracketing wildcards (e.g., where interim_date = "*10/15/86*").

3.1.3 Reduce the Use of Relational Joins

The second part of the solution was to reduce the use of relational joins wherever possible. Specifically, joins can be eliminated in the basic process of accumulating and displaying the logical record. Also, the effect of joins can be substantially reduced in the query process by retrieving only one field—the primary key; thereby limiting the use of joins to situations when the user specifies both keyword and master-record qualifying criteria.

To accomplish these objectives, the retrieval of logical records is divided into a two-part process: each user-specified query retrieves only the primary keys of qualifying logical records, then the primary keys are used to retrieve the contents of the master record and the related keyword records. The query portion of the retrieve is optimized because it needs to handle less data, and because it can be run only against the master table, or only against the keyword table. The data-accumulation portion of the retrieve is optimized because it relies exclusively on primary key retrieves—the fastest available.

A cost of this "split retrieve" approach is that it requires multiple accesses to the data base for a single functional retrieve. This would cause a problem if large groups of logical records had to be retrieved and displayed at one time: the retrieve request would be effectively doubled as the DBMS retrieved 25 primary keys, then retrieved the 25 logical records.

However, the split retrieve approach is feasible in the ACMS because the CFU's "record-keeping" and "reference" functional requirements dictate a need to view logical records one at a time, sequentially accessed—analogous to scanning pages in a book. Thus, the primary key retrieves are distributed over several screen changes in a single viewing session, and many of the qualifying records are not retrieved at all if the user finds the desired information in one of the first records viewed.

A second factor in the feasibility of the split retrieve approach is the programmability of EQUEL-based INGRES applications (other INGRES application tools[8] do not support split retrieves). Sets of primary keys are stored in FORTRAN arrays, then processed upon user action to retrieve the necessary logical records. Also, the implementation of split retrieve processing in a programmable environment offers other operational advantages, such as reporting counts of retrieval "hits," and giving users the flexibility in reviewing retrieved records (i.e., program-controlled retrievals either "forward" or "backward" through the primary key array).

---

[8] The other application development tools are comparatively easy to use, but make exclusive use of relational joins. Interestingly, these tools—which impose an inherent performance burden—are often pivotal in attracting users and developers to a particular relational DBMS, based on ease of use.

3.1.4 Other Alternatives Considered

A more traditional approach to reducing the use of relational joins is to avoid retrieving data from related tables. This approach is based on the premise that the content of a logical record (i.e., what data elements the user needs to access) depends on what the user is doing. An example of this can be found in a parts data base: if a user is reviewing part numbers and descriptions, there is no need to access pricing, procurement, or inventory data from related tables—and no need to invoke the implicit relational joins. Joins in such an example would be required only when the user explicitly wishes to review a particular subject area.

In the ACMS, however, the functional requirements dictated user-access to all data pertaining to a particular item of correspondence. Only one functional requirement—tracking of assigned responses—might be satisfied through access to a subset of the data elements, but the tracking function subsets are already set aside in reports and only one of the possible repeating fields would be used in the tracking function.

3.2 DATA BASE SIZE

The size of the ACMS data base also had a severe impact on performance. Specifically, any query that produced a sequential search (i.e., any search for a value in a non-key field) required approximately one minute of searching for every 3,000 logical records—on a completely unloaded VAX user running only the one test process. With a data base size of 50,000 records, this "ideal conditions" response time of 17 minutes was unacceptably long.

The solution to the problem of data base size was a function of dividing the ACMS data into manageable data sets, thereby reducing the number of records searched at any one time. Because of the nature of data usage, different mechanisms had to be developed for dividing older, "archives" data and more active, "current" data.

The division of data was done on the basis of a single attribute: "closing" of the correspondence item (e.g., sending a response, if a response was required). Archives data was divided into single-year data sets, based on the date closed. Access to archives data is provided by temporarily loading archives data (from tape, at user request) into a designated INGRES archives table. Current data was divided into "open" and "closed" data sets. The open/closed division leaves a fairly large "closed" data set, but the number of transactions against the much smaller open table was expected to represent the majority of transactions, thereby substantially reducing response times for "most" transactions.

3.2.1 Problem Conditions

The size of the ACMS data base was defined by past usage of the existing automated system. As of 1985, the total number of stored records was 50,000, and the annual rate of increase had been increasing steadily since the system had been installed in 1978. The most recent years' experience indicated an average annual rate of approximately 8,000 records per year, ranging from 6,000 to 10,000. Although modest increases in the rate of growth were considered possible, the average of 8,000 records per year was judged by users to be an accurate estimate, and comparatively stable.

Because of the frequency of "reference" requests for past correspondence, the user requirements for interactive access were determined to span 2 to 3 years of data (16,000 to 24,000 records).

Tests that were used to select an INGRES data base structure (see Section 3.4, below) also indicated that

retrieval times against a table containing 20,085 records could be as long as 7 minutes, even under "ideal" conditions. Specifically, the tests performed a range of searches (including exact matches, wildcards, different sized targets) against a non-key field in the master table. The tests were comprised of FORTRAN-based compiled programs (EQUEL), and were conducted with no other users on the VAX cluster. Response times ranged from 7:04 to 8:02 minutes (noted in minutes:seconds).

While these tests provided useful information for comparing different data base structures, they also revealed lower limits for response times. Experience indicated that loaded-system response times could be expected to be at least 2 to 3 times longer than the unloaded-system ideal times. Extrapolation suggested that the best-case response time of 21 seconds per thousand records would translate to an effective response time of 45 to 60 seconds per thousand. For a 50,000 record data base, the implied sequential search times of 40 to 50 minutes were judged unacceptable.

One additional factor served as a constraint on possible approaches: limited data storage on the VAX cluster. Specifically, the estimated 150 megabytes required for a 50,000 record data base (including application software) was almost double the amount that could be set aside for the application.

3.2.2 Implemented Solution

The solution was to divide the data base, thereby reducing the number of records to be searched at any one time. Mechanisms had to be developed for dividing the ACMS data into more manageable sets, then providing ad hoc access to any single set. Two different mechanisms were developed: one for "archives" data, one for "current" data.

3.2.2.1 Division of Archives Data

The archives division dealt with data that was older than the 2-to-3 year limit set by users for immediate access. This mechanism for division also had to be responsive to the data storage limit.

The ACMS archives data was divided into single year data sets, according to the value of the date_closed field (a correspondence item is "closed" when a response has been sent or, if no response was required, on the date received). A procedure was developed to remove archives data from the "current" tables on an annual basis, and the cutoff date for the date_closed field would be set so the remaining current records would include two years data. For example, on January 1, 1990, the archives for 1987 would be removed, leaving all ACMS data from 1988 and 1989. Throughout 1990, the amount of "current" data would increase (approaching three years' data), and on January 1, 1991, the archives for 1988 would be removed—again leaving two years' data.

The mechanism for providing access to archives data was to store all archives on tape, to establish an INGRES archives table pair ("amaster" and "akeyword," representing both master and keyword records), and to load individual archives from tape into the archives table pair upon user request. The application would employ a separate command for running search criteria against the archives table pair.

Once a desired archives year was on line, the data would be accessible within 6 to 8 minutes for sequential searches (based on an expected archive data base size of 8,000 records). If the desired archives had to be loaded from tape, there would be a delay of at least several hours. These conditions were judged by users to provide an acceptable tradeoff of archive coverage and working response times.

3.2.2.2 Division of Current Data

Division of current data was necessary because sequential search response times were predicted to range between 15 and 20 minutes (based on an expected 16,000 to 24,000 records in the "current" data base, and extrapolating the 20,000-record median with the predicted effective response time of 45 to 60 seconds per thousand records). The prospect of each user being able to complete only 3 or 4 searches against "current" data in an hour was deemed unacceptable for day-to-day access.

The division of ACMS current data was based on the same functional data component that was used for dividing archives data—the date closed. Current data, however, was divided not on the basis of *when* a particular item of correspondence was closed, but *whether* it was closed at all. This attribute produced a data division that combined small size and high usage, and offered users a high degree of consistency in anticipating and understanding the data divisions.

The open records were estimated to be a comparatively small subset of current data—probably 300 to 600 records. Disproportionately, the volume of transactions against open records was estimated to represent a substantial majority—probably 70 to 80 percent—of total update and retrieval transactions. (It should be noted that there remains a substantial number of updates against closed records, since the terms "open" and "closed" apply only to the progress of correspondence responses and not to data base activity. Some ACMS records are also appended directly as "closed" data.)

The mechanism for providing access to current data was to establish two separate INGRES table pairs (including both the master and keyword tables): one pair for closed records ("cmaster" and "ckeyword"), and one pair for open records ("omaster" and "okeyword"). The application would employ separate commands for performing searches against the closed or open tables, and a third command for searching both table pairs.

The tests that were used to select an INGRES data base structure (see Section 3.4, below) included tests against a 371-record table. Again, the tests were conducted under ideal conditions to compare data base structures, rather than predict loaded-system response times, but under those ideal conditions the sequential search against the 371-record table ranged from 9 to 10 seconds. Using unloaded-system to loaded-system experience, non-key access against the open table could be expected within 30 seconds—well within acceptable limits for that type of search. The size of the closed table, however, was not reduced substantially by removal of open records (accordingly, the predicted response times were not improved for closed data), but the resultant decrease in searches against that table diminished the expected occurrence and impact of 20 minute sequential searches.

To minimize the need for non-key searches, it was decided to define virtually every field subject to regular searching as a secondary key. Search patterns could not be measured on the existing system, but user-supplied estimates were used to identify 17 secondary keys. The keyword descriptor in the keyword table was also defined as a secondary key.

3.2.3 Other Alternatives Considered

Alternatives that were considered but rejected included different strategies for dividing current data, and approaches to improve the speed of the basic application development environment. Each of these alternatives is discussed briefly.

Different strategies for dividing current data were considered. Each alternative tried to define a comparatively small set of current data that represented the highest percentage of update and retrieval activity.

One approach would be to define a set of records that included all open records and all closed records that had been closed in the last 30 days. This alternative would substantially increase application complexity, and would increase the difficulty of predicting where a given record might be located. The latter result, in particular, would effectively eliminate the advantage of having a small file that can be searched rapidly—since users who do not have a good idea where a record is located would have to search both files.

A second alternative for dividing current data would be to group all records that had some form of data base activity (including append) in the last 30 days. This alternative might successfully define a set of records with the highest percentage of future activity (although it might not), but it completely eliminates the advantage of the exercise. Again, users would have very little chance of predicting where a given record is stored and the resulting usage pattern would likely be to search both parts of the data base—thereby eliminating the advantage of having a small file that can be searched rapidly.

The division of current data that *was* implemented was not accomplished without a cost in application complexity: all transfers between open and closed files must be done interactively, as the user changes the status from open to closed, or vice versa. It was difficult using EQUEL to track the progress of all steps in the combined append/delete operation, and to "back out" the effects of a partially completed transaction if a problem was encountered during update.

Another set of alternatives that was considered for resolving size-dependent performance problems dealt not with the volume of data, but with improving the speed of the application environment.

Several hardware/operating system-level alternatives for increasing data throughput were briefly investigated. These included dedicating a separate VAX host to the application; tuning the system parameters of one or more VAX cluster hosts to INGRES; increasing the priority of underlying INGRES processes; and installing a RAM-based disk emulator (the data base would be stored entirely in Random Access Memory (RAM), protected by an uninterruptible power supply, and backed up regularly; data throughput would be limited by the speed of the VAX bus rather than by hard disk data transfer rates). These alternatives were rejected because: a) they were considered beyond the scope of the project, and b) achieved response times were expected to be within acceptable ranges.

A software-based alternative that would modify the basic performance of the application environment was also considered. The suggested technique was to write a two-part custom "background processor" that handles all interaction between user programs and INGRES.

The first part of the background processor is a custom back end. User programs compose transactions to be applied to the INGRES data base (updates, deletes and appends); transaction requests are sent to the back end; the back end is interruptible for purposes of receiving and queuing transaction requests; the user program is free to compose additional transactions as soon as the transaction request has been accepted by the back end; and when not handling interrupts, the back end submits the INGRES transactions.

The second part of the background processor is a custom front end. The background processor starts each day by loading data base contents into memory—e.g., FORTRAN arrays; the arrays are made accessible to user programs; user programs conduct searches against the arrays instead of against INGRES data bases; and the arrays are updated by the back end.

The back end approach could be well suited to a transaction-intensive application where update transactions are very lengthy and there is a desire to free data entry operators from the idle time waiting for transactions to complete. The custom front end could help to provide very rapid access to a fairly small number of records (limited by VAX memory: 2,000 records would require 3.5 Megabytes of dedicated memory), but the complexity of writing a FORTRAN-based DBMS for access to FORTRAN arrays (INGRES does not support access to program arrays) could be quite cumbersome. In both cases, error handling and concurrency would be delayed substantially. These conditions were not well suited to the ACMS application and the background processor approach was not considered applicable to the ACMS size-related performance problems.

3.3 CHOICE OF AN INGRES DEVELOPMENT TOOL

INGRES' Application-By-Form (ABF) offers a framework for building a menu-driven application that invokes developed modules at user direction. There are several different languages, preprocessors, and other application development tools that can be used to develop the modules.

The actual mix of application development tools chosen was a function of many different factors, including maintainability, flexibility, features, user interface, and consistency. Performance was one of the factors considered in the selection process—and an important factor—but it was not determinative.

Very early in the system design process, it became useful to test comparative performance of two INGRES development tools: Query-By-Form (QBF) and FORTRAN-based Embedded Query Language (EQUEL). The results of those tests are presented in this section.

3.3.1 Selection Process and Test Methodology

In selecting application development tools, a concerted effort was made to use standard INGRES features—particularly features that are recommended in INGRES documentation. This was a strategy expressly developed to maximize consistency, maintainability, and portability of the application. These considerations also strongly discouraged use of non-INGRES development tools (such as programming languages and form-creation systems), non-INGRES system features (such as VMS files, VAX-specific FORTRAN library routines, or VMS system calls), or undocumented INGRES features (such as "tids"—tuple ID numbers).

What remained was to decide on the facilities that would be used to develop modules to be called in the ABF application. QBF represents INGRES' most automated development tool and can be used to rapidly develop most of the basic DBMS transaction handling and command structures (such as search, update, append, and delete). INGRES' Operations Specification Language (OSL) and EQUEL are programming languages that offer greater flexibility, but they require the developer to custom program all commands and operations. EQUEL was selected over OSL as a programming tool—based on programmability and feature limitations in OSL—and the FORTRAN-based version of EQUEL was selected over other versions (such as BASIC, COBOL, and C) on the basis of features, available user interface, ease of maintenance, and ease of modification.

In comparing EQUEL and QBF, it appeared that QBF had some severe operational restrictions. Rather than

expend considerable resources to identify the limits of QBF programmability, a series of performance tests was developed to determine whether QBF would be disqualified on performance grounds—without regard to programmability. With these limited objectives, the test did not warrant sophisticated control conditions or extensive testing. The tests are described below.

Nine query scripts were developed to operate against three test correspondence table pairs (see Section 3.3.2, below). The queries were run during normal business hours, when the VAX was being used for other purposes; there were usually between 10 and 15 other users; no other users were accessing the test data base or any part of INGRES software; monitoring of system process activities indicated that no large jobs were run during the tests. Queries that completed in under 5 minutes were repeated 10 times, queries that took between 5 and 10 minutes were repeated 5 times, and queries that took more than 10 minutes to complete were repeated only 3 three times. The results from repeated tests showed very little variability, and that variability decreased as the length of response increased. The pattern of results suggests that the decreasing variability reflected a flatter distribution of conflicting VAX processes. Queries that failed to complete within twenty minutes were terminated. Performance times were measured by hand (QBF does not have a facility that would permit system-based time tracking). Reported results are averages of recorded times.

Since we were not trying to test the performance impact of QBF's output and format limitations (one premise of the test was that mechanisms might exist to eliminate such limits), the EQUEL programs were constructed to emulate QBF output—displaying retrieved records on the user screen. The most significant impact of this choice was that comparative performance could be measured by recording the appearance of the first character of the first qualifying record. Since there were no perceptible delays of data transmission after that first character appeared, we concluded that measuring beyond that point would unnecessarily introduce data transmission variation into the results.

Table 1. Development Tool (QBF/EQUEL) Test Queries

Test Number	Field Size	Operator	Wildcards	53-Record Table Pair Count	53-Record Table Pair % of Total	371-Record Table Pair Count	371-Record Table Pair % of Total	20K-Record Table Pair Count	20K-Record Table Pair % of Total
Primary Key—Master Table									
1.	10	=	exact	1	2%	1	<1%	1	<1%
2.	10	>	exact	37	70%	57	15%	288	1%
Non-Key Field—Master Table									
3.	1	=	exact	35	66%	245	66%	13270	66%
4.	25	=	part*	1	2%	7	2%	379	2%
5.	25	>	part*	5	9%	35	9%	1893	9%
Non-Key Field—Keyword Table									
6.	20	=	exact	24	23%	168	24%	9107	24%
7.	20	>	exact	32	31%	217	31%	11733	31%
8.	20	=	part*	24	23%	168	24%	9107	24%
9.	20	>	part*	32	31%	217	31%	11733	31%

Table 2. Development Tool (QBF/EQUEL) Test Results
(minutes:seconds)

Test Number	53-Record Table Pair QBF	53-Record Table Pair EQUEL	371-Record Table Pair QBF	371-Record Table Pair EQUEL	20K-Record Table Pair QBF	20K-Record Table Pair EQUEL
Primary Key—Master Table						
1.	:14	:01	:24	:01	:24	:02
2.	1:13	:02	4:40	:24	15:20	13:06
Non-Key Field—Master Table						
3.	1:03	:04	2:11	:22	*	:44
4.	:12	:04	:43	:24	16:01	15:25
5.	:14	:04	3:33	:23	19:27	4:30
Non-Key Field—Keyword Table						
6.	:56	:02	6:41	:04	>20:00	:15
7.	:27	:01	9:03	:04	>20:00	:11
8.	:36	:01	4:40	:02	>20:00	:10
9.	:31	:01	6:01	:04	>20:00	:11

NOTES:

- Results reflect time to display first retrieved record. Time to display subsequent records are not shown, and did not vary significantly between QBF and EQUEL.
- Asterisk (*) indicates that this query caused the session to be terminated, disk allocation to be exceeded, and the data base to need restoration before further use.

The recorded times, therefore, represent only *part* of the actual retrieval time and cannot be compared to complete-retrieval times. In particular, they *cannot* be compared to the complete-retrieval times reported in Section 3.4.

3.3.2 Test Queries and Test Results

Queries were conducted against three separate pairs of related tables. Logical records were defined as master records uniquely identified with a 10-character alpha-numeric primary key, and repeated occurrences (no more than 10) of detail records related to the master by the same primary key. Master records contained approximately 1700 characters each; detail, or "keyword" records contained approximately 30 characters. The three table pairs are described below:

Table Description	Table Names	Number of Rows
53-record table pair	hmaster	53
	hkeyword	104
371-record table pair	tmaster	371
	tkeyword	707
20K-record table pair	cmaster	20,085
	ckeyword	38,288

The 9 sample queries used in the test are shown in Table 1. The test queries are grouped into three major categories:

- o primary key searches against the master table
- o non-key field searches against the master table
- o non-key field searches against the keyword table

Each test query is numbered. The table shows—for each "test number"—the length of the searched field and the query operator used to identify qualifying records. It also shows whether the data specified as criteria was an exact value, or a partial value used with the INGRES asterisk (*) wildcard character. When a wildcard was used, its relative position to the partial value is also shown. The table also shows a count of how many records from each table met the qualifying criteria, and what that count represents—expressed as a percentage of the searched table.

The test results, presented in Table 2, show the measured times to retrieve the first character of the first retrieved record, using the application development tool shown against each of the three table pairs. The "test numbers" correspond to the "test numbers" in Table 1. All times are displayed in minutes:seconds.

The results demonstrate rather conclusively that QBF, and the QBF-enforced relational join, could not be effectively used with the volumes of data required in the ACMS data base. EQUEL was therefore chosen for developing correspondence management modules.

3.4 CHOICE OF AN INGRES DATA BASE STRUCTURE

INGRES offers four basic data base structures; each structure is available in an alternate compressed form (the compressed data base structures employ algorithms to detect and remove frequently repeated characters, such as blanks, thereby reducing the amount of disk storage required to store the data):

Uncompressed	Compressed
HEAP	CHEAP
HASH	CHASH
BTREE	CBTREE
ISAM	CISAM

A discussion of how these structures operate is contained in INGRES documentation and training.[9] But the discussed characteristics—such as internal operation, dynamic index revision, and need for re-indexing—were subordinated to a single, measurable factor: performance. Under some circumstances these other factors might have considerable impact, but late-night processing availability at the VAX facility permits daily creation of all indices, and the aspects of internal operation were viewed as incidental if they did not translate into measurable performance advantages.

Tests were developed to compare performance of the available data base structures. The results of those tests are presented in this section.

3.4.1 Selection Process and Test Methodology

The purpose of the test was to compare performance of different INGRES data base structures against tables that would approximate the size and nature of the ACMS data.

In selecting which structures to test, a pre-screening process was used to reduce the number of tested structures to three, and the number of target table pairs to two.

It was not difficult to remove HEAP (and the compressed version, CHEAP) from consideration: HEAP is a non-indexed structure and promised an unacceptably long sequential search even when the uniquely-assigned control number was specified. The choice of testing compressed versions of the remaining structures—or uncompressed—was a difficult one. It was very likely that disk storage constraints in the production system would require use of compressed structures. However, we concluded that the algorithms for compressing and uncompressing were likely to be identical in each of the data base structures, and we did not want to add that factor into a test of comparative structure performance. This consideration was strengthened by the fact that we did not yet have reliable estimates of how much blank space would be in the data, and therefore did not yet know whether compression would be a large or small factor. Based on these considerations, the adopted strategy was to test the three indexed data structures in their uncompressed versions. The tests would be repeated using the compressed version of the apparent winner, to gain a better understanding of the potential effect of compression.

Among the three table pairs available and developed for testing (See Section 3.3, above), the 53-record table pair was not used. The 20,085-record table pair approximated the size of the ACMS "closed" data set; the 371-record table pair approximated the size of the ACMS "open" data set; but the 53-record table pair was not meaningful for any group of ACMS correspondence data.

A series of 31 EQUEL-based data base transactions was developed (see Section 3.4.2, below). Since the most important purpose of the tests was to compare performance of the structures—and not to predict actual user response times—the transactions were run during off hours, after completion of all backup procedures or on weekends, but only when the VAX was being used by no other users, batch jobs or other processing. All transactions were run through completion. Reported results were "wall clock" times, measured by access to FORTRAN-library VAX system calls ("lib$init_timer" and "lib$show_timer") in order to avoid INGRES-driven contention by using INGRES to access system statistics.

---

[9] Relational Technology, Inc., INGRES Reference Manual, Version 4.0 VAX/VMS, Alameda, CA, (1986).

3.4.2 Tests and Test Results

Transactions were conducted against two separate pairs of related tables. Logical records were defined as master records uniquely identified with a 10-character alphanumeric primary key, and repeated occurrences (no more than 10) of detail records related to the master by the same primary key. Master records contained approximately 1700 characters each; detail, or "keyword" records contained approximately 30 characters. The two table pairs are described below:

Table Description	Table Names	Number of Rows
371-record table pair	tmaster	371
	tkeyword	707
20K-record table pair	cmaster	20,085
	ckeyword	38,288

Each table pair was "modified" (an INGRES operation) to the tested data base structure on "control_no"—the primary key; a secondary key was defined for each master table, on the "act_office" field (used in the test transactions), and a secondary key was defined for each keyword table, on the "keyword" field (also used in the test transactions). All fields used in test queries were "optimized" for each table, a "sysmod" was run, and the tests were started immediately thereafter (the quoted functions are INGRES operations, recommended in INGRES documentation for maximizing performance). No operations were permitted to the data base between the start of the modify and the completion of the last test. The entire testing process lasted 5 to 10 hours. Interruptions were possible (occasionally unavoidable), and the tests were completed on another night or weekend.

The 31 sample transactions used in the test are shown in Table 3. The test transactions are grouped into five major categories:

- o primary key searches against the master table
- o non-key field searches against the master table
- o secondary key searches against the master table
- o secondary key searches against the keyword table
- o updates to the master table.

Table 3. Data Base Structure Test Transactions

Test Number	Operation	Field Size	Operator	Wild-cards	371-Record Table Pair Count	371-Record Table Pair % of Total	20K-Record Table Pair Count	20K-Record Table Pair % of Total
Primary Key—Master Table								
1.	retrieve	10	=	exact	1	<1%	1	<1%
2.	retrieve	10	>	exact	5	1%	288	1%
Non-Key Field—Master Table								
3.	retrieve	1	=	exact	245	66%	13270	66%
4.	retrieve	25	=	exact	21	6%	1134	6%
5.	retrieve	25	>	exact	182	49%	9850	49%
6.	retrieve	25	>=	exact	203	55%	10984	55%
7.	retrieve	25	=	*part	49	13%	2649	13%
8.	retrieve	25	=	part*	21	6%	1134	6%
9.	retrieve	25	>	part*	182	49%	9850	49%
10.	retrieve	25	>=	part*	203	55%	10984	55%
11.	retrieve	25	=	*part*	49	13%	2649	13%
Secondary Key—Master Table								
12.	retrieve	15	=	exact	42	11%	2282	11%
13.	retrieve	15	>	exact	294	79%	15912	79%
14.	retrieve	15	>=	exact	336	91%	18194	91%
15.	retrieve	15	=	*part	91	25%	4939	25%
16.	retrieve	15	=	part*	42	11%	2282	11%
17.	retrieve	15	>	part*	294	79%	15912	79%
18.	retrieve	15	>=	part*	336	91%	18194	91%
19.	retrieve	15	=	*part*	91	25%	4939	25%
Secondary Key—Keyword Table								
20.	retrieve	20	=	exact	77	11%	4160	11%
21.	retrieve	20	>	exact	7	1%	380	1%
22.	retrieve	20	>=	exact	84	12%	4540	12%
23.	retrieve	20	=	*part	77	11%	4160	11%
24.	retrieve	20	=	part*	77	11%	4160	11%
25.	retrieve	20	>	part*	7	1%	380	1%
26.	retrieve	20	>=	part*	84	12%	4540	12%
27.	retrieve	20	=	*part*	77	11%	4160	11%
Updates—Master Table								
28.	replace	1674	=	exact	1	<1%	1	<1%
29.	append	1674	=	exact	1	<1%	1	<1%
30.	delete	1674	=	exact	1	<1%	1	<1%
31.	replace	25	=	exact	1	<1%	1	<1%

Each test transaction is numbered. The table shows—for each "test number"—the operation performed the length of the affected field, and the query operator used to identify qualifying records. It also shows whether the data specified as criteria was an exact value, or a partial value used with the INGRES asterisk (*) wildcard character. When a wildcard was used, its relative position to the partial value is also shown. The table also shows a count of how many records from each table met the qualifying criteria, and what that count represents—expressed as a percentage of the searched table.

The test results are presented in two tables—Table 4 presents results against the smaller 371-record table, Table 5 presents results against the larger 20,085-record table pair. Both tables show the measured "wall-clock" times to complete the test transaction, when the table pairs were modified to the indicated data base structure. The "test numbers" correspond to the "test numbers" in Table 3. All times are displayed in minutes:seconds.

The results demonstrated that INGRES' BTREE data base structure produced the fastest retrieves against both ACMS master tables, particularly in sequential (non-key) searches against the large data set (approximately 4 minutes faster—33 percent—than any other structure). Update transactions in BTREE were not the fastest (ISAM-based updates against the large master table were approximately 2 seconds faster—25 percent), but since the performance "bottleneck" appeared to be sequential searches, retrieval response times were considered more important than updates.

For the keyword table, results did not indicate a clear choice: most results were within 10 percent of one another. Against the small keyword table, ISAM was fastest on six of eight tests. Against the large keyword table, ISAM produced the best results for the two longest tests but was slowest on all others. HASH was fastest on six of eight tests against the large keyword table. Again, since the significant bottleneck appeared to be in searching the large table pair, performance against the large keyword table was considered

Table 4. Data Base Structure Test Results—Small Data Set (seconds)

Test Number	INGRES Data Base Structures HASH	BTREE	ISAM
Primary Key—Master Table			
1.	.82	.99	.82
2.	21.24	.87	.72
Non-Key Field—Master Table			
3.	22.09	9.59	13.78
4.	21.23	9.40	13.63
5.	21.90	9.44	13.69
6.	21.87	10.53	13.72
7.	21.34	9.19	13.38
8.	22.84	9.80	13.66
9.	21.96	9.49	13.82
10.	22.17	9.77	13.73
11.	21.37	10.23	13.33
Secondary Key—Master Table			
12.	2.69	2.25	2.44
13.	15.15	10.76	12.62
14.	13.57	10.86	12.64
15.	21.78	9.87	13.36
16.	2.56	2.35	2.44
17.	13.49	10.93	12.69
18.	13.86	10.92	12.75
19.	21.75	10.15	13.41
Secondary Key—Keyword Table			
20.	2.15	1.81	1.85
21.	1.89	1.81	1.77
22.	2.05	2.18	1.98
23.	2.11	2.98	1.79
24.	2.02	1.84	1.87
25.	1.92	1.88	1.75
26.	2.06	2.37	1.95
27.	2.17	3.41	1.79
Updates—Master Table			
28.	2.44	4.53	1.68
29.	4.28	4.31	3.96
30.	1.15	.76	.50
31.	1.81	2.01	1.35

the most important—in particular, test number 20, since it was expected that most searching in the ACMS keyword tables would be done by exact match. The decision to use HASH, therefore, was based on superiority in six of eight large table tests, more consistent performance throughout the tests, and on superiority in the exact match test.

It should be noted that the relatively poor performance of HASH in the tested queries is considered primarily due to the use of non-key fields (test numbers 3 through 11), and to the use of partial values and comparative operators (test numbers 2, 13 through 19, and 21 through 27).

A final set of tests was conducted after modifying the test tables to the compressed versions of the selected data base structures: CBTREE for the master tables and CHASH for the keyword tables. The test methodology was the same as that described above; the results are shown in Table 6. The results confirmed that compression would not detract from performance in this particular test data base, and could actually deliver a substantial performance improvement.

4. IMPLEMENTATION-LEVEL FACTORS

This section describes the performance problems that were reported in the operational ACMS, the implementation-level factors that were determined to have a significant impact on performance, and the specific techniques and modifications that combined to resolve the performance problems.

Performance of the ACMS in the first few weeks of operation was marred by extreme line noise, severely interfering with data entry and retrieval. Response times were reported by users as being from 1 to 5 minutes—regularly exceeding 10 minutes, and occasionally in excess of one or more hours.

Several potential problem areas were identified, and simultaneous efforts were initiated to analyze and resolve them. Because of the severity of the problems, incremental solutions offering significant performance improvements were implemented immediately, and the chronology of implementation-level performance tuning was thoroughly intermingled.

Table 5. Data Base Structure Test Results—Large Data Set (minutes:seconds)

Test Number	INGRES Data Base Structures		
	HASH	BTREE	ISAM
Primary Key—Master Table			
1.	3.29	1.03	.86
2.	19:01.39	8.91	11.04
Non-Key Field—Master Table			
3.	18:57.06	7:48.28	11:50.28
4.	19:03.27	7:04.46	11:55.00
5.	18:51.69	7:28.10	11:55.74
6.	19:30.04	8:01.59	11:50.01
7.	19:01.07	7:43.70	11:55.97
8.	18:52.19	7:06.56	11:48.83
9.	18:57.03	7:27.75	11:47.37
10.	18:57.95	7:30.43	12:02.99
11.	18:46.85	7:42.98	11:54.85
Secondary Key—Master Table			
12.	1:29.41	1:05.65	1:24.91
13.	10:21.91	8:21.07	10:45.43
14.	10:41.76	8:10.48	11:00.75
15.	18:59.59	7:18.79	11:58.31
16.	1:24.98	1:07.12	1:24.59
17.	10:21.98	8:12.43	10:45.47
18.	10:35.54	8:32.65	10:55.02
19.	18:58.87	7:41.96	12:04.46
Secondary Key—Keyword Table			
20.	42.91	45.04	45.49
21.	44.54	48.02	50.38
22.	53.48	55.16	59.23
23.	1:43.43	2:06.12	1:23.21
24.	43.90	45.29	47.37
25.	44.89	48.14	50.71
26.	54.46	55.83	59.35
27.	1:49.17	2:05.52	1:19.36
Updates—Master Table			
28.	14.77	5.43	2.54
29.	8.75	4.88	4.61
30.	8.81	3.31	2.98
31.	13.10	1.73	1.42

Since the chronology and specific intermix of performance tuning implementation is only occasionally germane to the purpose of this report, this section will focus on the substance of the performance tuning factors and solutions, rather than on the sequence of events. The discussions of implementation-level factors are grouped into three functional areas:

- o Monitoring Operations and Performance
- o Application Environment Factors
- o Selection of Secondary Keys

All of the performance problems encountered after implementation were resolved through the performance tuning described in this section, and response times for most operations were lowered to within the 10- to 20-second objective defined at the beginning of the project. Observations and "lessons learned" from these performance tuning activities are summarized in Section 5.

4.1 MONITORING PERFORMANCE AND OPERATIONS

As problems were reported by users and incremental solutions were implemented, there needed to be a consistent mechanism for tracking the operation of the application and the effect of changes. The need for an operations and performance monitor also developed from some distinct advantages that could be realized in performance tuning in an operational rather than development environment. Specifically, the existence of live data and ongoing transactions offered valuable information about the ACMS data, data access and usage patterns, and actual (versus emulated or predicted) application performance.

INGRES does not offer a facility for tracking operations and performance, nor recommends a third-party monitor that would identify discrete transactions and report response times, CPU and disk usage, or other application-specific data. The solution was to develop a custom monitor by inserting measurement code into EQUEL-based application modules—which would append data reflecting operations and performance into INGRES-defined monitor tables. The monitor tables could then be reviewed as needed, providing a historical data base of monitor data, without significantly interfering with on-going ACMS operations.

Table 6. Data Base Structure Test Results—Compression
(minutes:seconds)

Test Number	371-Record Table Pair BTREE & HASH	371-Record Table Pair CBTREE & CHASH	20K-Record Table Pair BTREE & HASH	20K-Record Table Pair CBTREE & CHASH
Primary Key—Master Table				
1.	.99	1.05	1.03	1.44
2.	.87	.92	8.91	5.14
Non-Key Field—Master Table				
3.	9.59	5.35	7:48.28	4:38.68
4.	9.40	5.62	7:04.46	4:13.35
5.	9.44	5.48	7:28.10	4:24.75
6.	10.53	5.49	8:01.59	4:28.85
7.	9.19	4.27	7:43.70	3:46.50
8.	9.80	4.52	7:06.56	3:44.46
9.	9.49	5.60	7:27.75	4:39.17
10.	9.77	4.95	7:30.43	4:30.81
11.	10.23	4.31	7:42.98	4:15.89
Secondary Key—Master Table				
12.	2.25	1.69	1:05.65	38.18
13.	10.76	9.32	8:21.07	8:46.08
14.	10.86	9.26	8:10.48	8:01.45
15.	9.87	4.41	7:18.79	3:54.88
16.	2.35	1.57	1:07.12	33.40
17.	10.93	9.11	8:12.43	8:10.79
18.	10.92	12.26	8:32.65	8:48.13
19.	10.15	4.46	7:41.96	4:19.99
Secondary Key—Keyword Table				
20.	2.15	1.72	42.91	42.94
21.	1.89	1.67	44.54	44.19
22.	2.05	1.89	53.48	56.20
23.	2.11	1.88	1:43.43	1:22.02
24.	2.02	1.81	43.90	44.97
25.	1.92	1.69	44.89	44.21
26.	2.06	1.96	54.46	55.56
27.	2.17	1.88	1:49.17	1:23.99
Updates—Master Table				
28.	4.53	2.06	5.43	3.09
29.	4.31	3.95	4.88	4.51
30.	.76	.75	3.31	3.21
31.	2.01	1.49	1.73	1.84

As with the other post-implementation performance factors, the need for a performance monitor was addressed simultaneously with other factors and the facility was developed and implemented incrementally. There was incomplete information on what data was needed; the operations portion of the monitor was used in an application debugging process (which is not described in this paper); and the nature of required operations and performance data changed as different solutions were investigated. As the monitor was being developed, every effort was made to retain consistency of collected data—to permit meaningful comparison of data collected over time. However, some measurement techniques had to be changed as inherent biases or inaccuracies were discovered. The final version of the monitor was changed to consolidate all the partially implemented monitor components into a single facility that produced consistent, accurate, complete results, with maximum efficiency and flexibility.

The final version of the performance and operations monitor appends a single record to an INGRES-defined "monitor" table for each ACMS correspondence transaction. Cumulative system values are captured at the beginning and end of each transaction, and the record is appended with key data related to the transaction (including the difference between "start" and "end" values) and the transaction's effect. Data captured includes:

- user—INGRES "username" indicating VAX USERID
- date—INGRES "_date" including time of day
- module—name of EQUEL program in use
- operation—nature of ACMS operation
- table_pair—affected INGRES table pair
- effect—such as: success, failure or delayed
- cpu—INGRES "_cpu_ms" CPU seconds
- elapsed—INGRES "_bintim" wall clock time
- direct_io—INGRES "_dio_cnt" direct disk I/O count
- m_control_no—correspondence control number
- m_where_cl—where clause used in transaction (part)
- m_rcount—INGRES "rowcount" number of rows affected
- m_ingerr—INGRES "errorno" transaction return code

In addition, three alpha fields (named "a," "b," and "c") and three numeric fields ("x," "y," and "z") were defined, to permit ad hoc capture of transaction-related data without modifying the monitor.

Clearly, some portions of the monitor data (such as module, operation, and control_no) contained essential identifying information. Some portions of the monitor, however, proved to be more valuable than originally expected. In particular: the user field was valuable in tracking operational anomalies; the effect field permitted identification of transactions that had failed or been delayed through user action; and the elapsed field produced the most meaningful comparative performance statistic. The cpu and direct_io fields were used occasionally to isolate communications and secondary key problems, but were not especially useful in actual performance tracking.

The measurement of elapsed "wall clock" times proved to be the most difficult: INGRES' "_et_sec" produced dramatically unreliable results. The method implemented was to note the start and end times, expressed as the number of seconds since January 1, 1970 (INGRES "_bintim(0)" function), and capturing the difference. Elapsed times measured using this technique and actual times measured at user terminals were consistently within 2 or 3 seconds of one another.

Captured performance data was then available for any query that might be desired, but for purposes of regular operations and performance review the data was summarized into two key reports: response time distribution, and query field-usage.

To produce the response time distribution report used in daily review of transaction response times, the wall clock results from the monitor table were summarized and appended into a second INGRES-defined table ("freq") in which response times were broken down into the following distributions:

- 1st-level distribution (ACMS operation)
  - append
  - move
  - replace
  - retrieve
- 2nd-level distribution (response time intervals[10])
  - 0 to 5 seconds
  - 5 to 10 seconds
  - 10 to 15 seconds
  - 15 to 20 seconds
  - 20 to 30 seconds
  - 30 to 60 seconds
  - 60 to ** seconds

A representative sample of the response time distribution report is presented in Figure 3. Of the identified application operations, most are standard DBMS transactions (append, replace, retrieve); but the

---

[10] Response time intervals are measured as greater than the first value, and less than or equal to the second value. For example, the response time interval "10 to 15 seconds" includes responses of greater than 10 seconds, and less than or equal to 15 seconds.

* * * * Executing . . .

date	operation	seconds	frequency
09-mar-1987	append	0 to 5	
09-mar-1987	append	5 to 10	10
09-mar-1987	append	10 to 15	2
09-mar-1987	append	15 to 20	4
09-mar-1987	append	20 to 30	1
09-mar-1987	append	30 to 60	
09-mar-1987	append	60 to **	
09-mar-1987	append	--avg-----	10.588
09-mar-1987	move	0 to 5	
09-mar-1987	move	5 to 10	1
09-mar-1987	move	10 to 15	7
09-mar-1987	move	15 to 20	10
09-mar-1987	move	20 to 30	3
09-mar-1987	move	30 to 60	
09-mar-1987	move	60 to **	
09-mar-1987	move	--avg-----	16.048
09-mar-1987	replace	0 to 5	60
09-mar-1987	replace	5 to 10	2
09-mar-1987	replace	10 to 15	
09-mar-1987	replace	15 to 20	
09-mar-1987	replace	20 to 30	
09-mar-1987	replace	30 to 60	
09-mar-1987	replace	60 to **	
09-mar-1987	replace	--avg-----	1.758
09-mar-1987	retrieve	0 to 5	91
09-mar-1987	retrieve	5 to 10	28
09-mar-1987	retrieve	10 to 15	7
09-mar-1987	retrieve	15 to 20	1
09-mar-1987	retrieve	20 to 30	
09-mar-1987	retrieve	30 to 60	2
09-mar-1987	retrieve	60 to **	2
09-mar-1987	retrieve	--avg-----	7.023

Figure 3. Sample Response Time Distribution Report

"move" operation is specific to the ACMS application. Specifically, moves occur when the status of a correspondence item is changed from open to closed (or from closed to open) and the logical record is moved from the open table pair to the closed table pair (or vice versa). A recorded "move" operation, therefore, includes both an addition to one table pair and a deletion from the other.[11]

The distribution scale used in the response time distribution report is not an arithmetic progression—the size of each interval increases as the response times increase. These response time intervals were tailored to the particular performance history of this application and to the target performance objectives. Under other circumstances, an appropriate distribution scale could be very different.

A second key part of the monitor was the query field-usage report. This report, an example of which is presented in Figure 4, produced a cumulative record of the fields used in user-defined queries, together with the number of times each field was specified. This data provided usage pattern information that was very helpful in selecting secondary key fields (see Section 4.3, below). One of the system costs of running the field-usage part of the monitor is that a separate "monitor" record was appended for every field used in a query. Accordingly, a complex query could impose an overhead of 15 to 20 INGRES appends before the query was actually started.

```
* * * * * * Executing . . .

|s_field    |number       |
|-----------------------|
|act_inf    |            1|
|act_office |            3|
|clerk      |            1|
|control_no |         1525|
|corr_type  |            5|
|date_recd  |            4|
|date_writ  |          157|
|fp_number1 |            5|
|fp_year1   |            5|
|from_code1 |           99|
|from_code2 |            2|
|from_name1 |          117|
|from_name2 |            1|
|from_org1  |            6|
|from_org2  |            1|
|keyword    |          158|
|security   |            2|
|to_code1   |           92|
|to_name1   |          107|
|to_org1    |            1|
|topic      |            8|
|-----------------------|
(21 rows)
continue
```

Figure 4. Sample Query Field-Usage Report

4.2 APPLICATION ENVIRONMENT FACTORS

The operations and development environment of the ACMS application had a severe impact on performance, but these factors were generally outside the scope of performance tuning. They are reported here because of their significant potential impact on performance. The factors include: communications, ALL-IN-1, and software releases.

4.2.1 Communications Environment

As shown in Figure 1 (see Section 2.1, above), CFU users were connected to the VAX cluster through dial-up asynchronous modems, then connected through a switch to a user-selectable VAX cluster host ("node"). Line noise experienced by users was sporadic, and occasionally quite severe: in some cases, INGRES commands could not be executed because line noise would not permit entry of discrete commands.

Numerous efforts were undertaken to resolve the line noise problems—such as changing user modems, changing VAX modems, examining telephone lines, and direct connection to the VAX (bypassing the cluster switch). The noise abated somewhat, but persisted on occasion and was eventually found to be causing several operational anomalies: the user's terminal would have all the function key definitions changed; the telephone connection would be dropped; the modem would reset; data would be lost; and the screen would freeze, go blank, or change to a graphics character set. Examination of these problems—particularly the case where the screen would apparently freeze—eventually revealed that several reported "performance" problems were actually caused by communications problems.

Since the communications environment was not in our scope of control, we do not have statistics on the impact of specific changes. However, by the time the communications problems were resolved, performance times had been reduced substantially: most appends completed within 20 to 30 seconds; most non-primary key retrieves completed within 10 to 40 seconds.

Not all of the noted improvement was attributable to communications—several other environmental factors had changed and secondary keys had been tailored. But the pattern of dramatic performance problems decreased steadily as communications problems were incrementally resolved, and the patterns continued to decrease until the problems were only sporadic and readily traceable to individual transactions. Also, the nature of secondary key tailoring strongly suggests that communications had been responsible for: the 1-to-10 minute response times on key-field searches; on virtually any poor retrieval performance when the primary key (control number) was supplied; and on any responses beyond 10 minutes.

4.2.2 ALL-IN-1 Environment

As also shown in Figure 1 (see Section 2.1, above), the ACMS is invoked from within DEC's ALL-IN-1 Office Automation system, and operates as a sub-process under ALL-IN-1. The impact of operating the ACMS as a sub-process is not known.

The VMS operating system is specifically designed to offer the same processing environment—in terms of both performance and operability—for any VMS process or sub-process. However, that goal is not always achieved. ALL-IN-1, in particular, changes the sub-processing environment in a number of ways that we observed over the course of the project, and presumably in other ways that we did not observe. Also, the combination of INGRES and ALL-IN-1 produced some unpredictable results. The question was raised as to whether the combination of INGRES and ALL-IN-1 might be having a significant impact on performance.

[11] It should be noted that the usual DBMS "delete" operation is not shown because CFU users do not ordinarily delete correspondence records from the ACMS data base

The advantage of maintaining a consistent user interface for VAX operations was considered by VAX systems personnel to be very strong. Accordingly, no attempt to temporarily investigate the potential impact of removing the ACMS from ALL-IN-1 was made. Under other circumstances, however, this environmental factor might be further investigated.

4.2.3 Software Releases

A number of new software releases were implemented during the application development and performance tuning periods. This included two new releases of INGRES, a major release of ALL-IN-1, and an upgrade to VMS operating system. Performance was not affected by the non-INGRES upgrades; the INGRES upgrade from 3.0 to 4.0 occurred before reliable performance test procedures could report significant performance changes; and the upgrade from INGRES 4.0 to 5.0 is estimated to have produced a 10 to 15 percent performance improvement. Each of these upgrades and results is described below.

Following implementation of the ACMS, VAX VMS was upgraded from version 4.4 to version 4.5. At the same time, ALL-IN-1 was upgraded from version 2.0 to 2.1. The application had to be recompiled, but there was no significant impact on performance.

INGRES was upgraded from version 3.0 to 4.0 towards the end of application development activities. As mentioned, this was before test procedures had been developed that would reliably report performance changes. Operationally, however, no significant improvements were observed.

INGRES was upgraded from version 4.0 to 5.0 about a month after implementation. The INGRES Release Notes Version 5.0/02 (Relational Technology, Inc., 1986) states that "5.0 is primarily a performance release of 4.0," offering "significant performance improvements in ... queries involving btree tables." The installation immediately followed secondary key tailoring, and preceded resolution of communications problems.

For the few days of performance statistics that were collected surrounding the INGRES 5.0 installation but unaffected by other significant implementation-level changes, there was no reliably demonstrable performance improvement, but it seems quite likely that a performance improvement did occur. Specifically, average retrieval times increased from 13.714 seconds to 19.744 seconds and average append times decreased from 48.315 seconds to 33.500 seconds. However, significant communications-related problems were still occurring that could have distorted average results. When times under 100 seconds are removed from the results, the retrieves under version 5.0 show a 27.53 percent improvement (from 10.652 down to 7.720) and appends show a 19.85 percent improvement (from 30.633 down to 24.553). These improvements, however, are likely to be overstated. Actual performance reflects varying loads on the VAX cluster and the three days measured after the new release was installed were the three days before the Thanksgiving holiday: overall VAX usage was noticeably lower. Our estimate is that the new release of INGRES actually improved performance 10 to 15 percent.

4.3 SELECTION OF SECONDARY KEYS

The selection of secondary keys was the single most significant implementation-level factor that was within proper control of the performance tuning project team. Secondary keys are essential for rapid access to a large data base—thereby reducing the need for searching the data sequentially. But the existence of secondary keys causes response times for appends and updates to increase substantially—as indices are updated. The most important factor in secondary key tuning is to find the particular set of secondary keys that produce the optimal mix of retrieval and update performance. In the ACMS, usage statistics identified six fields that represented 98 percent of all searches. Secondary keys were defined for each such field and the update response times were able to be brought into an acceptable range (see Section 4.4.1, below).

The initial set of secondary keys defined in the ACMS data base included 17 fields. Response times for appends and updates were verified to be running as long as 5 to 10 minutes—occasionally as long as 20 minutes—without the effect of communications problems. Disk I/O statistics indicated the presence of extraordinarily high disk activity, suggesting that the number of secondary keys was responsible.

Secondary keys were then eliminated from the ACMS file structures. The effect on performance was a dramatic improvement of response times for appends and updates, and a dramatic degradation of response times on all queries (except when the primary key was specified). The result was that users concentrated their activities on appends and updates, and use of the system for retrieval and reference purposes was limited almost exclusively to instances in which the control number was already known. While this process successfully improved reported response times (appends and updates were completed in 20 to 30 seconds), the situation would not be acceptable on a long term basis.

The field-usage portion of the custom monitor (see Section 4.1, above) captured the information essential to meaningful secondary key tailoring: which fields were being used in user queries?

Initially, secondary keys were defined on the basis of user interviews and claimed requirements. As the monitor's field-usage statistics developed a significant history, six secondary keys were identified. The accumulated query field-usage report presented in Figure 5 shows that the six fields—together with the primary key—represented more than 93 percent of all user queries, and that all of the six fields were used substantially more often (generally more than ten times as often) than the next most frequently used fields.[12]

As additional fields were added (one at a time), an attempt was made to judge the performance impact of each change. However, differences in daily work loads —by CFU users, by other users on the VAX, and by other system developers—contributed to a wide variance of performance results. There was no single secondary key that improved response times by an amount that could be reliably measured from actual-use performance data, but the cumulative effect of the secondary keys was both clear and dramatic.

When all six secondary keys were implemented (but before all the communications problems were resolved) most appends and updates were completing in under 60 seconds. 98 percent of all queries were being conducted with the use of secondary keys. Queries that were fairly straightforward, single-field specifications were completing within 20 to 30 seconds; queries that used complex combinations of wildcards, comparative operators (greater than or less than), and master-keyword joins were taking 1 to 5 minutes; and the 2 percent of non-

[12] It should be noted, however, that incremental implementation of secondary keys must be regarded as having introduced a possible bias towards keys that were defined earliest in the process: user experience could have lead to greater usage of those fields, since they were producing faster results. But because of the wide disparity of usage patterns, it appeared that the effect of such bias was slight and that the selection of secondary keys reflected actual usage requirements.

```
* * * * * * Executing . . .

|s_field   |number       |          Cumulative Percent of Total
|-------------------------|          (not part of standard report)
|control_no|         1525|                ----------------
|keyword   |          158|                  66.30 %
|date_writ |          157|                  73.17 %
|from_name1|          117|                  80.00 %
|to_name1  |          107|                  85.09 %
|from_code1|           99|                  89.74 %
|to_code1  |           92|                  94.04 %
|topic     |            8|                  98.04 %
|from_org1 |            6|                  98.39 %
|corr_type |            5|                  98.65 %
|fp_number1|            5|                  98.87 %
|fp_year1  |            5|                  99.09 %
|date_recd |            4|                  99.31 %
|act_office|            3|                  99.48 %
|from_code2|            2|                  99.61 %
|security  |            2|                  99.71 %
|act_inf   |            1|                  99.80 %
|clerk     |            1|                  99.84 %
|from_name2|            1|                  99.88 %
|from_org2 |            1|                  99.92 %
|to_org1   |            1|                  99.96 %
|-------------------------|                 100.00 %
(21 rows)
continue
```

Figure 5. Query Field-Usage Report—Two Months Cumulative Data

key searches often required 5 to 15 minutes. When the communications problems were resolved, these responses settled into the following predictable ranges:

- o appends: 15 to 20 seconds
- o moves: 15 to 25 seconds
- o replaces: 20 to 30 seconds
- o retrieves: 5 to 15 seconds

These results do not include the 1-to-3 second response times for retrieval when the control number was specified, nor the occasional complex-query search that ranged from 30 to 90 seconds, nor the occasional sequential search that required 5 to 10 minutes.

These times were within the 10- to 20-second response time targets established at the beginning of the project. Additional tuning, specific to programming in INGRES and not within the scope of this paper, further improved performance of the replace operation. Following the INGRES-specific tuning, the average response time for the replace operation was under 2 seconds.

5. SUMMARY

The principal objective of performance tuning the ACMS application was to bring response times to the interactive level described during the initial stages of system development: "pre-defined system operations must be accomplished within 10 to 20 seconds." This objective was achieved.

The performance tuning process produced several findings that are valid only for the particular circumstances of the ACMS application, running in the then-current hardware and software environment of the agency's VAX cluster, based on the user requirements, data usage patterns, and operational preferences of the CFU personnel. However, many of these findings may be relevant to applications with similar characteristics, such as (depending on the nature of the specific problem encountered): similar correspondence management functional requirements, VAX or VAX-cluster operating environments, an INGRES development environment, or a development environment using a relational data base.

This section describes findings from the ACMS performance tuning efforts, and describes the general conclusions that might be drawn from these findings.

5.1 FINDINGS FROM ACMS PERFORMANCE TUNING

The most significant performance improvements in the ACMS were attributable to the following factors:

- o Abandoning the "relational" data base model was essential. Relational joins had to be virtually eliminated, even if it meant modifying the logical record structure to violate the first-normal-form principle of the traditional relational model (to reduce the number of related tables). The process of only retrieving primary keys from user-specified queries, then using primary key searches to collect and accumulate logical records (the split-retrieve approach) was also important in reducing the use of relational joins.

- o Division of data was also essential. Under the existing configuration of hardware and software, the application simply could not produce satisfactory response times against the entire 50,000-record data base.

- o The selection of secondary keys produced a significant performance improvement. In the ACMS application, some number of secondary keys had to be used to gain rapid access to the volume of on-line data, but a high number (17) of secondary keys degraded performance too much. Measurement of post-implementation usage provided the most reliable information for selecting the final set of six secondary keys.

- The operations and development environment also had a dramatic impact on performance. For the ACMS application, these factors included: communications, selection of the most efficient development tools and structures, and effective use of available DBMS programming techniques.

- Implementation of a performance monitor was an essential component to the performance tuning process. In this case, the monitor had to be developed and installed in the INGRES application on a custom-programming basis.

The key finding that pertains to performance tuning in a relational environment, is that *for certain applications* the relational model (and the relational join, in particular) contains an inherent characteristic that *prevents* reasonable performance—irrespective of the hardware, the DBMS, the operating system, or any other factor affecting implementation of the relational model software. Specifically, the multiplicative effect of the relational join produces a situation in which the number of records retrieved in order to accumulate a single logical record is extraordinarily large. In the example application, there were 7 detail records in each of 7 detail tables; the resulting join produced a retrieve that had to go through 823,543 records. With approximately 1750 characters per record, the total amount of data that had to be processed was 1.44 gigabytes—*per logical record*. Adherence to strict relational principles, and to strict normalization techniques, ensures a high number of related tables in even a moderately complex application. If the user requirements dictate access to all related data—enforcing the concept of the logical record—then the result will be a system in which performance is measured in days, rather than seconds.

5.2 CONCLUSIONS

Although the actual findings presented in this report apply only to the individual circumstances of the ACMS, the nature of the problems identified and resolved suggest that the following conclusions might apply to data base application performance tuning in general:

- The relational data base model is often regarded as capable of emulating virtually any data base structure. However, the performance implications of emulating other structures—especially if several relational joins are required—suggest that while emulation may be possible, it might not be practical, even with a very small data base (e.g., 53 records). The key to this performance limitation is de-normalization during relational joins: de-normalization creates a multiplicative effect of related detail records stored in related tables, which can be devastating to performance.

- Performance tuning requires concerted effort in evaluating and choosing data set characteristics: division of data into discrete tables; DBMS "structure" of each table; and secondary keys. The solution to each of these problems may require exploration of a wide range of alternatives, thorough pre-implementation testing, and a willingness to make dramatic changes even after the system is operational, but each has a significant impact on performance—especially when the size of the data base is large.

- A thorough familiarity with the features and application development tools of the selected DBMS is essential during the performance tuning process. This does not mean that all possible features must be used—in fact, many offer only marginal performance improvements or may introduce unwarranted complexity into the application. But understanding the DBMS features and tools is important in identifying possible alternatives, and in evaluating their potential performance-related and design-related impact.

- When concerted post-implementation performance tuning is required, a good operations and performance monitor is especially important, even if it means that the monitor must be custom-coded. Monitoring an operational system can also provide information that is extremely valuable in performance tuning—information that cannot be determined through user interviews or simulation.

A final factor that was extant throughout the ACMS performance tuning process, but which was not a deliberately considered approach with alternatives and analysis (and therefore not subject to analysis in this report), was a positive working relationship between the CFU users and the performance tuning project team. Without this relationship, without the communications, without the openness, candor and hard work that characterized the relationship, the performance tuning effort would have suffered dramatically. Coordination of tuning efforts with the users and good communications was essential to being able to tune an operational system, and to achieving user satisfaction.

EMPIRICAL MODELS OF DASD RESPONSE TIME

J. B. Major
IBM Canada Ltd.
Montreal, Quebec

ABSTRACT

Several models of system-wide average DASD response time are studied. Factors considered are: utilization of the average and of the busiest actuator as well as the time-varying nature of these, that of the average DASD controller and the system-wide average DASD service time. The latter is, in turn, modeled with respect to actuator types and the presence or absence of cache controllers. DASD response time is represented as a family of piece-wise linear lines for specific levels of DASD controller utilization. The results are used to stipulate DASD configurations that would give specified response times for desired levels of DASD traffic. Model validation is reported for various groupings of the more than 100 days' data that were contributed by more than 20 installations in a wide range of industries of mostly MVS/XA systems, many with shared DASD. Interpretation in terms of some commonly used queueing models is discussed.

INTRODUCTION

In the course of his work with several clients the author has collected hundreds of hours' worth of DASD performance data. Attempts were made to relate I/O response times (RT) as reported to workload and I/O device related information, data that could be obtained from the Direct Access Device Activity report of the Resource Measurement Facility (RMF) of MVS/SP2. It is noted that actual RT may exceed the reported level due to software queueing. Resource usage and I/O response time relationships included considerations for different I/O device types (IBM 3380 type disks, standard, or D and E models, IBM 3350 type disks and IBM 3880 or equivalent storage controllers with or without cache).

Response time models of these types of environments have been offered before (cf. <1>, <2>, <3> and <4>, as examples). All of the models are based on probabilistic and queueing considerations and necessitate unverifiable assumptions about the nature of the I/O traffic. Simulation might be offered as a validation tool. Unfortunately, the simulation runs are burdened by essentially equivalent and unverifiable assumptions about the I/O traffic with the reassuring result that the mathematical formula and the discrete event simulation both predict comparable response times with remarkable accuracy as compared to one another. Accuracy with respect to the real world is another matter.

The work reported here gives I/O response time predictions usually within 6% of the observed figures when the observed I/O service time (ST) is considered to be given. In fact, ST itself has to be predicted. The proposed prediction of ST is accurate usually within 7% of the measured results. Prediction errors of this size might be considered more than acceptable. However, when one considers the configuration implications of these errors, one might have another view. The point is discussed in the Section on PREDICTION ACCURACY.

The models proposed require estimates of system-wide DASD I/O rates (per second) during the planned period of operation (usually the prime shift). In addition, we need the system-wide average connect time (per I/O), as well as the expected SKEW of the I/O load. SKEW is defined as the maximum DASD actuator busy level divided by the average DASD actuator busy level over all the actuators of a configuration. This idea has been motivated by <5>. RMF reports the connect time by actuator and by Logical Control Unit (LCU). The system-wide average can be calculated from the LCU data. Finally, the fraction, FNCH of the non-cached DASD Control Units (CU) as well as the fraction, FNDE of the non-D-or-E 3380 actuators is required to complete the estimation of ST. One can view these fractions as configuration rules. No assumption is made about the ST distribution or about the I/O interarrival times.

Measurements of more than 20 systems over more than 100 shifts in a variety of industries validate the predictions.

This paper is a generalization of the author's study of a Law of Diminishing Returns (LDR, first stated in <6>, and refined in <7> with respect to the I/O traffic capacity of DASD actuators as seen in the Section on LDR PERSPECTIVE.

To facilitate reading, acronym definitions are given in FIGURE 11.

TRANSACTION RESPONSE TIME

DASD I/O response time is an important consideration in the process of configuration selection to the extent that it influences the response time of transactions from the point of view of the users of a host system. Naturally, the user's perception of response time is generated by several factors: communication time, software caused delays, and, last but not least, I/O servicing. The latter includes all the code execution and processor contention times that are associated with servicing a "typical" I/O as well as the response time of the I/O subsystem. Thus, transaction response time is composed of

a. communication delay
b. initial software delay (e.g. swap-in or attach)
c. software serialization or synchronization delay
d. several I/O servicing instances (DASD and CPU)

Of these, a. is a function of workstation attachment (on local or remote, and, possibly quite slow communication channels) and is not discussed any further here. Items b. and c. are software implementation dependent and are not discussed here beyond the remark that synchronization is usually an application code function (to ensure data integrity), serialization may be caused by application code or by application control software (CICS, IMS, others). In the latter case judicious choice of environment setup parameters can significantly reduce or altogether eliminate these delays. The remaining terms relate to the operation of the processor (that is analyzed in a companion paper <8>) and to the DASD subsystem, the subject of this paper. Tape and communication I/Os are omitted altogether.

This view assumes that I/O processing of a transaction is not overlapped with I/O processing or with CPU processing of the same transaction instance, and that all I/Os caused by a transaction add to its response time. That is not entirely the case. Swap-in or swap-out I/Os of TSO transactions are usually processed in parallel to the extent of swap dataset and I/O path multiplicity. Similarly, I/O buffer write-out may be more or less parallel in IMS systems at the time of transaction termination and might not add to the transaction response time at all. Thus, when an I/O response model consistently overestimates transaction response times at the host system level, and the model has been verified to be fairly accurate for some on-line systems, one can suspect overlaps of the above kind. Analysis of some controlled measurements of TSO environments that the author had access to indicated an I/O overlap factor of 1.05 - 1.18,

meaning that TSO response time would be overestimated by that factor without allowance for overlaps. This factor includes delays of the b. and c. type, that is the actual overlap effect is really greater than 5 - 18 %. Similar study of IMS measurements resulted in underestimation of the average response time by a factor of 1.0 - 1.2, where the high values correspond to high processor utilization levels, and in 1.0 - 1.5 for CICS environments. Again, large serialization adjustments correspond to high processor utilizations. Details are discussed in <8>.

As an example of DASD I/O processing impact on transaction response time, a TSO transaction mix might receive an average response time of 900 ms on an IBM 3090 processor (model 180 and up) half of which would be accounted for by the DASD subsystem's handling of the 20 or so I/Os that are normally associated with such transactions. These numbers assume locally attached workstations and 3380 D or E type DASD with no cache controllers.

MODEL REQUIREMENTS

A predictive model is a formula or procedure that incorporates one or more variables whose values can be measured or stipulated and one or more constants that apply to all the predictions. Calibration consists of choosing the constants that give a satisfactory (in accepted terms) prediction for the measured data space. Validation with additional data should not degrade or improve the quality of predictions. Further, the constants should not change with changes in software and hardware usage or with the application mix, unless the model is geared specifically to a subset of conditions that are assumed to be constant over time. In this latter case we could have several models, each with its own formula or procedure, and with its own constants. This paper limits the context to MVS/SP systems with most of the environments studied being of the SP2 (XA) type.

The model should be validated in terms of actual practical installation data that cover a broad range of systems and that span many hours of operation. It should produce reasonable configuration recommendations. A desirable feature of a response time prediction model is its interpretability in terms of queueing concepts.

The DASD I/O response time model that is proposed here meets the above requirements.

DASD I/O RESPONSE TIME FACTORS

The empirical DASD I/O response time model is based on information normally available from RMF DASD Activity reports (in MVS/XA). The relevant items are:

B	average actuator utilization (% busy)
X	maximum actuator utilization (% busy)
X/B	actuator utilization SKEW
LB	average DASD LCU utilization (% busy)
PB	average DASD I/O path utilization (% busy, PB = LB/2)
S	DASD SIO rate (per second)
RT	average DASD I/O response time (ms)
ST	average DASD I/O service time (ms)
CT	average DASD I/O connect time (ms)
D.TRF	DASD traffic (ms/s) D.TRF = S•ST
P.TRF	DASD path traffic (ms/s) P.TRF = S•CT
RT/ST	normalized DASD I/O response time
N	number of DASD actuators multiple of 4 when type is 3380
NCU	number of DASD control units a CU usually contains 2 DASD paths
NCUCH	number of DASD control units with cache
FNCH	fraction of non-cache control units FNCH = (NCU - NCUCH)/NCU
NADE	number of 3380-D or E type actuators
FNDE	(N - NADE)/N
NA50	number of 3350 type actuators
FR50	NA50/N

The last 7 items are not reported by RMF. They can be obtained from configuration diagrams or knowledge. Further, RMF does not report system-wide data for shared DASD. Report consolidation is necessary. A report consolidation and analysis program was developed by the author's colleague, L. Beauchemin and used extensively in the validation process.

DATA ANALYSIS AND RESPONSE TIME MODELS

Consider the measurement scatter illustrated on FIGURE 1. As seen, there is no clear relationship between the RT/ST ratio and the average actuator busy %, B, other than, by and large, higher levels of B correspond to higher levels of RT/ST. The different symbols used represent measurements with different levels of X/B. The highest ratios (x's) represent values around 10. Dots correspond to ratios around 7.5, triangles to 6 and squares to 4. With that in mind, a schematic shown on FIGURE 2 can be suggested as a model of the RT/ST versus B relationship. This family of piece-wise linear lines that correspond to given fixed SKEW levels forms the SKEW Model.

The above measurements represent shift averages taken over one to five shifts per point. Similar scatters are obtained when each point represents one shift only.

Three models of RT/ST are discussed now.

1. The SKEW Model

General algebraic formulation of the model is obtained from FIGURE 3 and is as follows.

Let SLOPE = (R2 - R1)/(B2 - B1). Then

$$RT/ST = \begin{cases} R1 & \text{when } B < B1 \\ R1 + (B - B1)\cdot SLOPE & \\ ? & \text{when } B > B1 + (R3 - R1)/SLOPE \end{cases} \quad (1)$$

R1, R2, R3 and B1 are model parameters that are estimated from data analysis. For a specific SKEW the SLOPE is found as

B2 = B.MAX/SKEW

B.MAX is also a model parameter that is estimated from data analysis. Based on the data analyzed, the following parameter values were chosen:

R1 = 1.2, R2 = 1.4, R3 = 1.6, B1 = 2, B.MAX = 60

These give RT/ST estimates with average absolute error of prediction of 6 % with error defined as

error = 100•(prediction - actual)/actual

In these terms, (1) applies when

1 < SKEW < 30, and 2 < B < 2•(60/SKEW - 1)

Otherwise, it is assumed that when B < 2 then RT/ST = 1.2, and when B > 2•(60/SKEW - 1), then RT/ST > 1.6 and is not very predictable. In terms of these parameter values, SLOPE = 1/(5•(60/SKEW - 2))

2. Path Loadings: the Calibrated SKEW Model.

RT/ST varies as the DASD path loadings vary. Generally, the busier the DASD paths, the higher is RT/ST. The SKEW Model's estimate of RT/ST can be corrected by stretching (multiplying) by the factor of

SF = 1 + 0.01•(PB - 8),

The corrected prediction is

(RT/ST)' = (RT/ST)•(1 + 0.01•(PB - 8))

Very low (< 8 %) path loadings correct the RT/ST estimate to a somewhat lower level, whereas high path loadings increase the estimate. Use of this correction factor has reduced the average absolute error of the prediction by somewhat more than 0.5 %.

In summary, the calibrated SKEW model is given by

$$RT/ST = \begin{cases} 1.2 \cdot SF & \text{when} \quad B < 2 \\ (1.2 + (B - 2) \cdot SLOPE) \cdot SF & \\ 1.6 \cdot SF + ? & \text{when} \quad B > 2 + 0.4/SLOPE \end{cases} \qquad (2)$$

$$SLOPE = 0.2/(60/SKEW - 2)$$

3. The Aggregate Server Model

If the average I/O response time of device n of an N device system is RTn for an average I/O rate (/second) of Sn then the system-wide average I/O response time RT is obtained from

$$RT = \sum_{n=1}^{N} (RTn \cdot Sn) / \sum_{n=1}^{N} (Sn)$$

Imagine a fictitious aggregate server (DASD actuator) that has the same average response time RT, and has an average utilization B that is the average utilization of the N DASD actuators. This fictitious aggregate server and its waiting line can be considered to form a single server queue. The Pollaczek-Khinchin (P-K) mean value formula (cf. <9>) could give a solution to this queue.

The emphasis is on capacity planning: represent the DASD system through a small set of parameters that yield equivalent response time to that of N DASD servers, each with its own queueing behavior without regard to the idiosynchrasies of specific actuator-dataset-access method combination effects.

Detailed models of DASD systems represent every actuator and its behavior through their various approximations. These could be P-K models, actuator by actuator. Average DASD response time for the system is obtained from accumulation of the individual actuator response times.

The aggregation view is capacity planning motivated. In contrast, individual device models address performance recommendations.

The P-K model, aggregate or individual, assumes that I/O requests come at random to be serviced by a server (the actuator) whose service time distribution has a squared coefficient of variation of C. It is frequently referenced as the M/G/1 queueing model. The estimation task is that of finding a C value that results is acceptable prediction error on RT/ST. In general terms,

$$RT/ST = (1 - B \cdot (1 - C)/200)/(1 - B/100)$$

Based on the author's data, good aggregate predictions are obtained for C = 10. In that case

$$RT/ST = (100 + 4.5 \cdot B)/(100 - B)$$

When 1.2 < RT/ST < 1.6, the formula yields very nearly a straight line. FIGURE 4 illustrates the fit. This model is somewhat less accurate than the SKEW Model and has an average absolute error of 8 %, approximately. Apart from its lower predictive accuracy, this model has conceptual difficulties: it is very hard to justify C = 10 as the service time descriptor. In fact, estimates of data analyzed by the author indicate C = 1.5 or so (<8>, APPENDIX II). Further, I/O request interarrival times are not random. Whatever is the distribution, its mean varies hour after hour as the workload intensity varies. Calibration of C to fit with a specific installation's data is possible if one stipulates constancy of C for that installation for lengthy time periods (several months as a minimum) to make this approach workable.

The last difficulty with the P-K model is the fact that the number of concurrently active I/O operations targeted to any one actuator is always very finite whereas the model is that of an open queue, that is one with an infinite source. To remove this difficulty, one would have to use an M/G/1/T model with a maximum number, T, of concurrently active I/O tasks. The range of 1 - 6 for T would be practical for most individual device models while a somewhat higher number could be argued for the aggregate server model. The conceptual problem changes to stipulating a reasonable multitasking level, and justifying the assumption about an even larger C than that of the open P-K model. The latter case is even less defensible than the original problem. Numerical solutions can be obtained for any non-negative C and positive integer T by following the developments of <10>.

Just as the SKEW model necessitates the inclusion of path loadings logic, the M/G/1/T model requires some underlying logic to represent these loadings. Finally, there is the problem of time-varying average server utilization. The latter is discussed further in the QUEUEING INTERPRETATION Section along with an interpretation of the plausibility of the C = 10 assumption.

SERVICE TIME MODEL

All of the above response time models assume some given ST. In fact, ST is a variable that needs to be predicted from more primitive data. Having analyzed over 20 installations and many shifts of ST data the following estimator has been developed for ST and is used in the CONFIGURATION EXAMPLES Section:

$$ST = 2 + 1.6 \cdot CT + 8 \cdot FNCH + 3 \cdot FNDE + 4 \cdot FR50 \qquad (3)$$

It states that the ST difference between two systems, one with all cache, the other one with no cache is 8 ms, on average. Consideration is given to the fact that some number of I/Os require full DASD service (read miss and write I/Os). Specific representation of detailed cache behavior is not attempted. Rather, aggregate system behavior is studied and is approximated.

Further, the ST difference between XA supported D and E type 3380s and standard type 3380s is 3 ms, on average. Finally, 3350 type actuators add 4 ms to ST over and above their longer connect time contribution (3 ms•1.6 = 4.8 ms).

The formula has been validated with data gathered by the author as of at the time of writing and was found to have an average absolute error of 7 %. It is noted that the 3350 presence in the configurations measured was very small. As a result, ST estimates with significant 3350 presence might be less accurate than stated. Graphical illustration of the relationships is not readily possible due to the presence of four independent variables in the definition of ST.

Equation (3) attempts to give a simple relationship, not necessarily one with the smallest possible error. The average service time impact of other DASD technologies could be similarly approximated once performance experience becomes available from a sufficient number of installations.

RESPONSE TIME STABILITY

The average DASD I/O response time, RT is a random variable as is the average DASD I/O service time, ST. Responsiveness of two systems with the same RT but with different ST may be perceived differently by their respective users even when all other conditions are identical. The one with higher RT/ST has more variability (less predictability) of its DASD response with concomitant undesirable fluctuations in its overall transaction response time. Thus, high RT/ST should be avoided.

When faster I/O devices are introduced (with much reduced ST) some increase in RT/ST can be tolerated as the resultant RT is also much reduced. Compromises are necessary, though. As an example, halving ST should not be accompanied by doubling RT/ST, that is by preserving the same RT as before. Rather, one should aim for a reduced (not necessarily halved) RT and for a somewhat increased RT/ST. Put differently, new technology should not be a mere replacement for the old one, rather, it should be an improvement - the story of data processing during the last three decades.

LDR PERSPECTIVE: GENERAL COMMENTS

To quote from <6>: "... a system with 2N like components cannot do twice the work of one with N like components of the same type." A quantification of this law of diminishing returns was stated in <6> and refined for actuators in <7>. It is restated below.

DASD N	Expected D.TRF (ms/s)	DASD N	Expected D.TRF (ms/s)
8	1100	28	3500
12	1640	32	3920
16	2130	40	4700
20	2600	48	5360
24	3070	60	6100

when N > 60, D.TRF = 2500 + 60•N

Another view of this phenomenon was expressed in <5> through the idea of SKEW. McNutt defined SKEW as the highest observed (on some actuator) I/O rate divided by the I/O rate of the "average" actuator. He showed that this SKEW increased with the size of the actuator pool: the more actuators in the DASD pool, the fewer I/O per second are processed by the average actuator.

The current analysis examines SKEW in terms of traffic (ms/s), rather than in terms of I/O rates. As in <5>, the SKEW increases with the DASD pool size. The author's data leads to the approximate relationship

SKEW = 4.2 + 0.02•N +- 2 (4)

The observation scatter is shown on FIGURE 5. The relationship is weak but visible (correlation is 0.7).

The LDR as stated in <6> and <7> addresses the question: how many actuators are needed to handle a specific level of DASD traffic with "reasonable" (1.3 < RT/ST < 1.4) responsiveness. The Calibrated SKEW Model leaves the definition of "reasonable" to the user and addresses the question: how many actuators are needed to handle a specific level of DASD traffic with a specific DASD response time, for specific DASD path loadings. The algebra is given below. Examples of its use are discussed in the next Sections.

When B is between the limits 2 and 2 + 0.4/SLOPE, that is between 2 and 2•(60/SKEW - 1), it is expressed from (2) through the relationship:

B = 2 + 5•((RT/ST)/SF - 1.2)•(60/SKEW - 2) (5)

with SF, the stretching factor defined as

SF = 1 + 0.01•(PB - 8)

Outside of these limits the RT/ST value is impractical for the commonly encountered DASD configurations.

APPLICATION: THE CONFIGURATION TASK

A typical configuration task would entail finding the number and type of actuators and the number and type of DASD CU's for given S (DASD I/O rate per second) and RT, CT, responsiveness targets (low and high RT/ST) and SKEW target. One could examine different configurations for various PB levels.

N is obtained from the DASD traffic definition:

D.TRF = S•ST = 10•B•N

or,

N = S•ST/(10•B) (6)

This N value is used to verify the SKEW target's likelihood of attainment by using (4). If the target is between the calculated limits it is considered achievable. Otherwise, an N' is calculated for a probable SKEW level of 4.2 + 0.02•N, a sometimes necessary step.

CONFIGURATION EXAMPLES

Configure a DASD system with RT = 18 ms and RT/ST = 1.35 that will handle 400 I/Os per second. This and the other examples are originated in the companion paper <8>. It is expected that CT = 5 ms and SKEW = 7. These CT and SKEW assumptions match the validation data closely (FIGURE 10). Consider 3380 type D and E DASD only.

Based on the specifications ST = 18/1.35 = 13.33. To apply (3) we observe that FR50 = 0 and FNDE = 0. Now, ST = 2 + 1.6•5 + 8•FNCH = 13.33. That gives FNCH = 3.33/8 = 0.42, i.e., 58 % of the DASD CUs should be cache controllers.

Total DASD traffic is D.TRF = 400•13.33 = 5332 ms/s. Total DASD path traffic is P.TRF = 400•5 = 2000 ms/s.

Consider various path busy levels. Explore DASD paths for PB = 20 %, 16 % and 12 %, respectively. That necessitates 2000/200 = 10, 2000/160 = 12.5 and 2000/120 = 16.7 DASD paths, respectively. In round numbers we need 5, 6 or 8 DASD CUs, 3, 4 or 5 of which should be cache controllers, respectively.

Apply now (5) to calculate B for different PBs.

B = 2.18 when PB = 20 %
B = 3.64 when PB = 16 %
B = 5.22 when PB = 12 %

Application of (6) gives the N values, respectively:

N = 5332/21.8 = 244 (to the nearest multiple of 4)
N = 5332/36.4 = 148 (to the nearest multiple of 4)
N = 5332/52.2 = 104 (to the nearest multiple of 4)

FIGURE 6 summarizes this and 3 other examples, the latter ones with somewhat modified conditions. FIGURE 7 is another set of 4 examples, each with S = 480, and otherwise similar to the first set. The SKEW target of 7 leads to configurations most of which when reevaluated by applying (4) end up with probable SKEW estimates plus or minus 2 units within the target. These configurations are, therefore, accepted as plausible ones. The others are recalculated for the nearest integer SKEW value.

Another set of calculations is performed in FIGURE 7 for the assumption that SKEW = 4 is possible to achieve. Finally, the original LDR guidelines are used to estimate the number of actuators required for the various D.TRF levels. A comparison can now be made between the various ways of configuration approaches.

Example details are clarified through the following notes.

• High RT/ST gives less stable response than lower RT/ST for equal average response times.

• Low path busy is usually a good idea: increased CU cost is offset by greater reduction in actuator cost. However, watch out for exceptions.

• Probable skew is obtained from the formula:

SKEW = 4.2 + .02•N

Recalculation of N (actuators) is done using (5) and (6).

• Note that not all the calculated configurations are practical. Normally, a maximum of 32 actuators would be configured on a Control Unit. As a result, configurations with N > 32•CUs are rejected.

ACTUATOR TYPE MIXING: THE HALF AND HALF CASE

Other configurations can be analyzed, as well. Consider a case with 50 % D/E and 50 % STD type 3380 actuators, an extension of the examples shown on FIGURE 7. This is the "half and half" case, H/H.

The service time calculation according to (3) gives

ST = 8 + 8•FNCH + 1.5 + 2 = 11.5 + 8•FNCH

If RT = 18 ms, RT/ST = 1.35 then ST = 13.33 ms and FNCH = 0.23, while if RT = 29 ms, RT/ST = 1.35 then ST = 21.48 ms and FNCH = 1.00. To cover the possibilities of FIGURE 7, we get

Case	H/H	H/H	H/H	H/H
RT (ms)	18	18	29	29
RT/ST	1.35	1.50	1.35	1.50
ST (ms)	13.33	12.00	21.48	19.33
8•FNCH	3.33	2.00	8.48	6.33
FNCH	.23	.06	1.00	.98
PB = 20 %				
CU's	6	6	6	6
with cache	5	6	0	0
PB = 16 %				
CU's	8	8	8	8
with cache	6	8	0	0
PB = 12 %				
CU'S	10	10	10	10
with cache	8	9	0	0

The numbers of DASD CUs and actuators are the same as those on FIGURE 7, the numbers of cached controllers are different due to the differences in the calculated FNCH values. More cache controllers are required than in the D/E case to compensate for half the actuators being the slower STD type. Also, fewer cache controllers are required than in the STD case, representing the improved responsiveness of half the actuators (the D/E types).

To avoid the tedium of applying (5) and (6) to several configuration possibilities, the N values have been tabulated for selected PB, SKEW and RT/ST levels in FIGURE 9, the Generalized LDR for actuator traffic, for a D.TRF of 10000 ms/s. Other traffic levels proportionally increase or decrease these N values. As an example, when D.TRF = 5332, SKEW = 7 and RT/ST = 1.35 the PB = 20 % table shows that

N = 460•5332/10000 = 244

Note that the DASD LDR suggests N = 125 actuators for D.TRF = 10000 ms/s (N = (10000 - 2500)/60 = 125). FIGURE 9 highlights the N values that are (approximately) 125 to indicate the SKEW, PB and RT/ST levels at which the SKEW Model and the LDR Model meet.

ACTUATOR TYPE MIXING: VARIABLE PROPORTIONS

Now, assume that half the controllers are cached. Following the examples of FIGURE 7 we obtain ST = 14 + 3•FNDE and see, readily, that the RT = 18 ms and RT/ST = 1.35 or 1.50 requirements cannot be met. Suppose that we aim for RT = 23 ms and RT/ST = 1.35 or 1.50. Now, the ST targets are 17.03 and 15.33. That gives FNDE = 1.0 and 0.4, respectively. Investigate now the condition that S = 512 I/O per second and PB = 16 %.

In the same manner, one can stipulate S, PB, RT, CT, FNDE and FNCH with SKEW and RT/ST targets to end up with a variety of possibilities. As an example, let PB = 16 %, RT = 20 ms, S = 512 I/O per second, CT = 5 ms and FNCH = 0.25 with approximate targets of RT/ST = 1.5 and SKEW = 7.

Now, P.TRF = 512•5 = 2560 ms/s. When PB = 16 %, 8 CUs are required, some with cache. The calculations are summarized below.

Option	1	2	3	4	5
S	512	512	512	512	512
RT	20	20	20	23	23
SKEW	7	7	7	7	7
FNCH	0.25	0.25	0.25	0.50	0.50
FNDE	0.75	0.50	0.25	1.00	0.40
ST	14.25	13.50	12.75	17.03	15.33
RT/ST	1.40	1.48	1.57	1.35	1.50
D.TRF	7296	6912	6528	8719	7849
N at 10000	194	133	97	275	122
N at D.TRF	142	92	63	240	96
3380 D/Es	36	44	48	0	56
3380 STDs	104	48	16	240	40
CUs	8	8	8	8	8
with cache	6	6	6	4	4

A comparison of Options 1 and 3 shows 64 actuators (mostly D/E type, Option 3) accomplishing the work of 140 actuators (mostly STD type, Option 1) at the same RT but at somewhat higher (less stable) RT/ST. Option 2 is a compromise solution. Similarly, in Option 5 the 56 E type actuators can pick up the work of 200 STD type actuators (Option 4) provided that the data stored on the 200 fits on the 56 (if not then up to 100 might be required) at the cost of some instability in RT/ST but without degradation of the average RT.

LDR PERSPECTIVE: SPECIFIC COMMENTS

The 1.3 - 1.4 RT/ST ratio originates from pre-MVS/XA environments. The pre-XA versions of RMF understate DASD RT in that software queueing of page/swap I/O requests is left out of the reported DASD RT (the reported DASD queue length, to be precise). As a result, the real RT/ST ratio is more like 1.4 - 1.5 when paging is present to any significant degree instead of 1.3 - 1.4.

Application of the LDR formula recommends 125 actuators for a DASD traffic of 10000 ms/s ((10000 - 2500)/60). Examination of FIGURE 9 shows that this number coincides with the recommendations of the Generalized LDR when RT/ST is in the range of 1.4 - 1.5, SKEW is in the range of 5 - 7 and PB is in the range of 12% - 20%.

These parameter values are encountered fairly frequently in actual installations, although, RT/ST is often in the range of 1.3 - 1.4, even in MVS/XA systems. However, the system-wide average DASD busy is generally < 8%, the level that the 125 actuators average to according to the LDR recommendations for a traffic of 10000 ms/s.

For low actual SKEW levels the Generalized LDR suggests smaller DASD configurations than those suggested by the LDR for RT/ST in the 1.4 - 1.5 range. When the SKEW levels are high, the Generalized LDR suggests more generous configurations than those calculated from the LDR formula. The two can converge after dataset placement tuning, an activity that reduces the SKEW.

PREDICTION ACCURACY

As indicated earlier, application of (2) has resulted in 5.5 % average absolute error in estimating RT/ST. Also, application of (3) has resulted in 7 % average absolute error in estimating ST. Error is computed as

100•(prediction - actual)/actual

The low error % of the RT/ST prediction is deceptive. Clearly, RT > ST for non-zero B values. Thus, one should estimate RT - ST or (RT - ST)/ST. Again, if (2) is used, we would have a 25 - 30 % average absolute error on (RT - ST)/ST, a far less impressive result. Unfortunately, other "precise" prediction models have the same problem. In consolation, the predictive accuracy of ST given by (3) is relatively good. The quantity predicted by this model is all of ST, with no large positive base to mask the huge errors of predicting an increment.

Consequences of the inaccuracy can be seen in the examples of FIGURE 6 and 7. Small changes in path loadings (PB) result in large changes in the number of required actuators. Small errors of RT/ST prediction when magically corrected would result in huge adjustments in the number of configured actuators. Viewed differently, fairly large under-configuration errors would result in relatively small performance (RT/ST) degradation. Conversely, generously configured actuator subsystems would perform only marginally better (in terms of RT/ST) than their skinny counterparts. In a tight budget environment it may be wise to err on the downside, that is to configure to a nominal specification of, say, RT/ST + 0.05 when the real target is RT/ST. FIGURE 8 shows a restatement of the example shown in FIGURE 7 in these terms. This is, effectively, a change of the ST and, therefore, the D.TRF estimate to 3 - 4 % lower values - a deliberate mistake, while leaving the RT target unchanged. The result is increased RT/ST, reduced D.TRF and reduced number of actuators. As seen, a 0.05 change in RT/ST (a 3 - 4 % change) results in 20 - 40 % change in the computed number of actuators - indeed important when a large number is recommended in the first place.

Under-configuration errors can result in minor performance degradation. Over-configuration errors will improve performance only marginally. When the data space capacity requirements of the DASD pool necessitate over-configuration, that pool will be a somewhat better than usual performer. If large data-capacity DASD is available, it is generally a good idea to configure this type of DASD at the risk of minor performance degradation when the prediction errs on the wrong side, a case that would be observed half the time.

In conclusion, RT/ST is not influenced much by DASD configuration size.

VALIDATION

Formula (2) and (3) were calibrated with data from 15 installations, each with one - five prime shifts of operation. Additional data pushed the number of installations to over 20 with more than 1200 hours of operation. Prediction error magnitude did not change with the new data. A set of data that was used in <5> and that spanned 15 minute intervals did, however, change the prediction accuracy, for the worse. These measurements featured very large SKEW values and normal (average of 4 %) DASD busy levels.

Model application to subsets of data did not result in too much change for the prediction errors. Groupings were formed as follows:

G1. daily observations, all systems;
G2. daily observations, shared DASD systems;
G3. daily observations, unshared DASD systems;
G4. summary observations, all systems;
G5. summary observations, shared DASD systems;
G6. summary observations, unshared DASD systems.

FIGURE 10 summarizes the validation results.

QUEUEING INTERPRETATION

Comments were made earlier on the inconvenience of both the M/G/1 and the M/G/1/T models. The main problem is that the average load is a random variable both in terms of time within a shift and device address within a pool of devices.

Consider a hypothetical case (also explained in <7>) of 13 actuators over a specific period of time during which 12 of the 13 have their B = 5 % and the 13th has its B = 50 %. Thus, the average of the 13 has its B = 8.5 %.

Assume that ST = 25 ms and that ST is independent of B (not true, in general). Now, the 12 actuators would process 2 I/Os each per second while the 13th would process 20 I/Os per second.

Assume, further, that RT/ST of an actuator is determined by an M/M/1 model (an M/G/1 model with C = 1): RT/ST = 1/(1 - B/100).

When B = 5 %, we get RT/ST = 1.05;
when B = 8.5 %, we get RT/ST = 1.09;
when B = 32.5 %, we get RT/ST = 1.48;
when B = 50 %, we get RT/ST = 2.00.

The average response time of the 12•2 + 20 = 44 I/Os per second is (24•1.05•25 + 20•2•25)/44 = 37.05, giving RT/ST = 1.48 - the same as that of an actuator with B = 32.4 %.

The pool of 13 actuators with average B = 8.5 % and with skewed I/O load (SKEW = 50/8.5 = 5.9) behaves the same way as a pool of perfectly tuned (SKEW = 1) actuators would in which every actuator is 32.4 % busy - a customary statement about how busy actuators can be in a "well-tuned" system. Thus, if individual actuators of an installation could be accurately described by the M/M/1 model (they cannot be) and if their DASD SKEW were approximately 6 (not unusual) this system at B = 8.5 % could be expected to look normal.

It is interesting to note that <2> reports a study of 9090 actuators (56 installations) with average B = 8.5 % for the overall pool.

Examine now the P-K mean value formula once more. We have

$$RT/ST = (1 - B\bullet(1 - C)/200)/(1 - B/100)$$

or, more conveniently,

$$RT/ST = 1 + (1 + C)\bullet B/(200 - 2\bullet B)$$

Hence,

$$C = (RT/ST - 1)\bullet(200 - 2\bullet B)/B - 1$$

If the aggregate RT/ST = 1.48 and if the aggregate response time is predicted accurately by the P-K formula then C computes to 9.33, not far from the 10 mentioned in discussing the aggregate server model earlier. Further, if the M/M/1 model above is replaced by an M/G/1 model, we obtain

when B = 5 %, RT/ST = 1 + 0.0263•(1 + C), 24 I/Os;
when B = 50 %, RT/ST = 1 + 0.5•(1 + C), 20 I/Os.

The average aggregate RT/ST is

$$RT/ST = 1 + (24\bullet 0.0263 + 20\bullet 0.5)\bullet(1 + C)/44$$

$$RT/ST = 1 + 0.2416\bullet(1 + C)$$

When C = 1, we get RT/ST = 1.48, as before.

Let C be the squared coefficient of variation of the service time of the individual actuators, Ca be that of the aggregate server. A relationship can now be calculated between C and Ca as follows.

$$1 + 0.2416\bullet(1 + C) = 1 + 8.5\bullet(1 + Ca)/183$$

The right hand side is RT/ST for the aggregate P-K server at B = 8.5 %. Therefore, when Ca = 10, we get

$$C = (0.5109 - 0.2416)/0.2416 = 1.11$$

When C = 1.11, we get RT/ST = 1.51. If every single actuator had the same RT/ST = 1.51, their utilization would have to be B = 32.6 %.

For the example, when all the individual actuators have C = 1.11 the aggregate server has Ca = 10 - the approximation referenced earlier.

SUMMARY

A DASD configuration approach to satisfy response time requirements has been proposed. Validation has shown that the I/O service and response time predictions are quite accurate. Nevertheless, small errors in response time prediction lead to large swings in the size of the actuator pool to be configured. When one deliberately overshoots the response time target by 3 - 5 %, a realistic configuration is obtained that will perform almost on target even when the response time prediction errs on the wrong side.

Configuration swings are minimized when the DASD pool is made up of faster type devices and is aided by cache controllers. In that case one can afford to design to relatively high response time to service time ratios and work in the area of low swings as a function of responsiveness prediction errors.

The configuration approach is a generalization of the author's earlier work concerning a law of diminishing returns relative to the performance of systems of many like components.

ACKNOWLEDGEMENTS

The author wishes to thank his colleague, L. Beauchemin for his help in developing the validation procedure of this paper, as well as for his comments on the material as it went through various phases of exposition. In addition, thanks are due to R. Wicks, G. Galambos, J. Ryden and L. Duyanovich for their helpful comments that shaped the paper to its present form.

REFERENCES

<1> "DASD Path and Device Contention Considerations", GG22-9217-00, March 1981, S.E. Friesenborg, available through IBM branch offices.

<2> "DASD Expectations: The 3380, 3880-23, and MVS/XA", GG22-9363-02, August 1985, S.J. Friesenborg and R.J. Wicks, available through IBM branch offices.

<3> "Modeling DASDs and Disk Caches", A. Brandwajn, Proceedings, CMG'86, December 1986.

<4> "Measuring I/O workload characteristics in IBM/MVS systems", G.E. Houtekamer, Proceedings, CMG'86, December 1986.

<5> "An Empirical Study in Variations of DASD Volume Activity", G. McNutt, Proceedings, CMG'86, December 1986.

<6> "Processor, I/O path, and DASD configuration capacity", J.B. Major, IBM Systems Journal, Vol. 20, No. 1, 63-85 (1981).

<7> "Capacity Planning with a SIO Methodology", J.B. Major, Proceedings of SHARE 65, 1985.

<8> "Empirical Models of Processor Response Time", J.B. Major, to appear.

<9> "Queueing Systems", L. Kleinrock, John Wiley and Sons, 1975.

<10> "On a stochastic process concerning some waiting line problems", L. Takacs, Theory of Probability and its Applications, Vol. 2, No. 1, 1957.

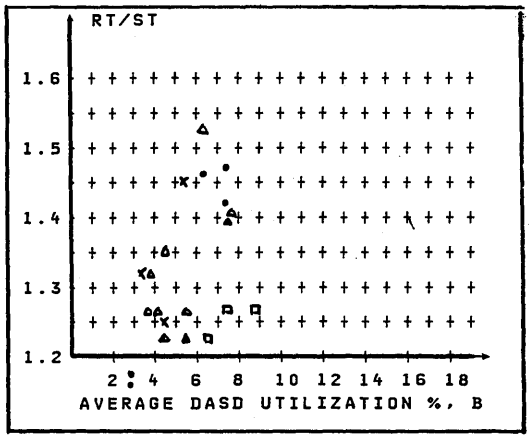


FIGURE 1 : DASD response measurements

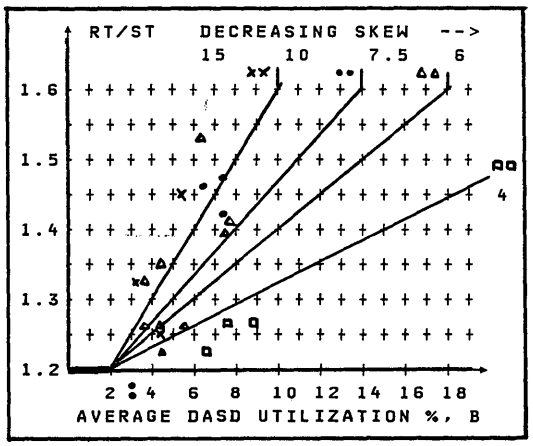


FIGURE 2 : The SKEW model of DASD response

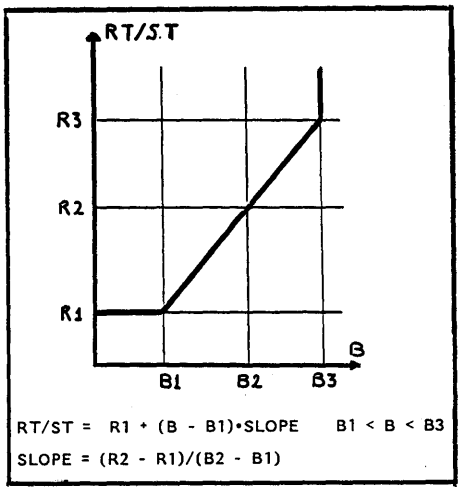


FIGURE 3 : The general SKEW model

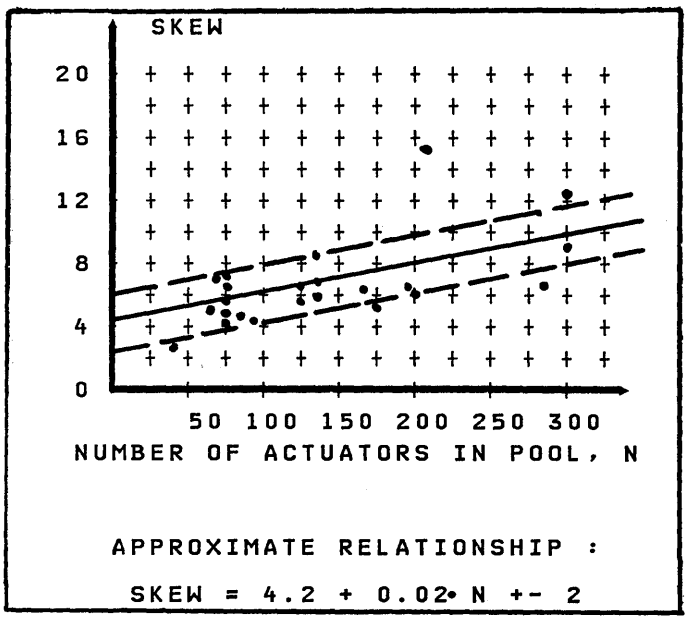


FIGURE 5 : DASD pool size and SKEW

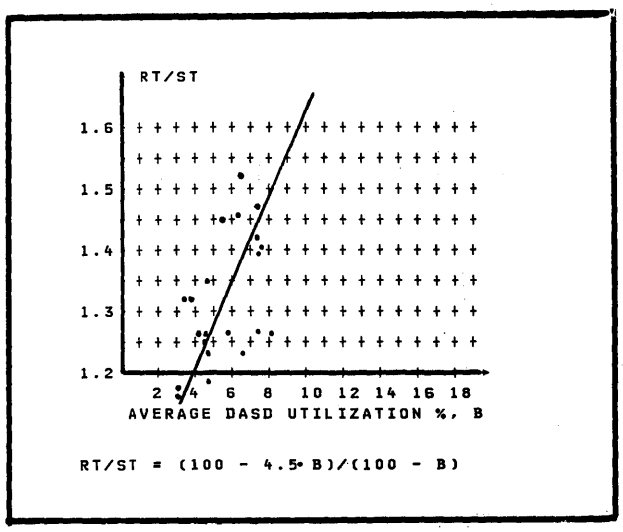


FIGURE 4 : The P-K mean value model

DASD configuration examples

Configure a DASD system to process

400 I/O per second, CT = 5 ms,
for average RT = 18 ms or 29 ms

ST = 1.6•CT + 8•FNCH + 3•FNDE + 4•FR50 + 2

D/E: Consider 3380 - D, E only: FR50 = 0
D/E: FNDE = 0, ST = 10 + 8•FNCH
STD: Consider standard 3380 only: FR50 = 0
STD: FNDE = 1, ST = 13 + 8•FNCH

Target RT/ST = 1.35 or 1.50
Target SKEW = 7

Case	D/E	D/E	STD	STD
S (/second)	400	400	400	400
CT (ms)	5	5	5	5
RT (ms)	18	18	29	29
RT/ST	1.35	1.50	1.35	1.50
ST (ms)	13.33	12.00	21.48	19.33
8•FNCH	3.33	2.00	8.48	6.33
FNCH	.42	.25	1.00	.79
P.TRF	2000	2000	2000	2000
D.TRF	5332	4800	8592	7732
PB = 20 %				
CU's	5	5	5	5
with cache	3	4	0	1
Prob SKEW	9.1	5.6	12.1	6.5
Actuators	244	72	396+	116
PB = 16 %				
CU's	6	6	6	6
with cache	4	5	0	1
Prob SKEW	5.8	5.1	6.8	5.6
Actuators	148	60	236	96
PB = 12 %				
CU's	8	8	8	8
with cache	5	6	0	2
Prob SKEW	5.8	5.0	6.8	5.6
Actuators	100	48	164	76

+ recalculated for nearest integral SKEW value.

FIGURE 6 : DASD Configuration example

DASD configuration examples

Configure a DASD system to process

480 I/O per second, CT = 5 ms,
for average RT = 18 ms or 29 ms

ST = 1.6•CT + 8•FNCH + 3•FNDE + 4•FR50 + 2

D/E: Consider 3380 - D, E only: FR50 = 0
D/E: FNDE = 0, ST = 10 + 8•FNCH
STD: Consider standard 3380 only: FR50 = 0
STD: FNDE = 1, ST = 13 + 8•FNCH

Target RT/ST = 1.35 or 1.50
Target SKEW = 7

Case	D/E	D/E	STD	STD
S (/second)	480	480	480	480
CT (ms)	5	5	5	5
RT (ms)	18	18	29	29
RT/ST	1.35	1.50	1.35	1.50
ST (ms)	13.33	12.00	21.48	19.33
8•FNCH	3.33	2.00	8.48	6.33
FNCH	.42	.25	1.00	.79
P.TRF	2400	2400	2400	2400
D.TRF	6398	5760	10310	9278
PB = 20 %				
CU's	6	6	6	6
with cache	4	5	0	1
prob SKEW	10.1	6.0	13.7	7.0
Actuators	304+	88	500+	140
PB = 16 %				
CU's	8	8	8	8
with cache	5	6	0	2
prob SKEW	7.7	5.6	9.8	6.4
Actuators	176	72	344+	112•
PB = 12 %				
CU'S	10	10	10	10
with cache	6	8	0	2
prob SKEW	6.7	5.3	8.0	6.0
Actuators	124	56	192	92
SKEW target	4	4	4	4
NA, PB=20	272	52•	440	84
NA, PB=16	120	40	196	64
NA, PB=12	76•	32	124•	52
LDR choice •	64	52	130	112

+ recalculated for nearest integral SKEW value.

FIGURE 7 : DASD Configuration example

DASD configuration examples

Configure a DASD system to process

480 I/O per second, CT = 5 ms,
for average RT = 18 ms or 29 ms

ST = 1.6•CT + 8•FNCH + 3•FNDE + 4•FR50 + 2

D/E: Consider 3380 - D, E only: FR50 = 0
D/E: FNDE = 0, ST = 10 + 8•FNCH
STD: Consider standard 3380 only: FR50 = 0
STD: FNDE = 1, ST = 13 + 8•FNCH

Target RT/ST = 1.40 or 1.55
Target SKEW = 7

Case	D/E	D/E	STD	STD
S (/second)	480	480	480	480
CT (ms)	5	5	5	5
RT (ms)	18	18	29	29
RT/ST	1.40	1.55	1.40	1.55
ST (ms)	12.86	11.61	20.71	18.71
8•FNCH	2.86	1.61	7.71	5.71
FNCH	.36	.20	.96	.71
P.TRF	2400	2400	2400	2400
D.TRF	6173	5573	9941	8981
PB = 20 %				
CU's	6	6	6	6
with cache	4	5	0	2
prob SKEW	7.5	5.6	9.7	6.4
Actuators	168	68	332+	112•
PB = 16 %				
CU's	8	8	8	8
with cache	5	6	0	2
prob SKEW	6.6	5.3	8.0	6.0
Actuators	120	56•	192	92
PB = 12 %				
CU'S	10	10	10	10
with cache	6	8	0	3
prob SKEW	6.0	5.2	7.2	5.6
Actuators	92	48•	148	80
SKEW target	4	4	4	4
NA, PB=20	116	40	188	64
NA, PB=16	76•	32	120•	52
NA, PB=12	52•	28	88	44
LDR choice •	64	52	124	108

+ recalculated for nearest integral SKEW value.

FIGURE 8 : DASD Configuration example

PB = 12 (Average DASD Path Busy %)

RT/ST = SKEW	1.30	1.35	1.40	1.45	1.50	1.55	1.60
3	154	92	66	51	42	36	31
4	190	119•	87	68	56	48	42
5	222	145	107	85	71	61	53
6	250	169	127•	102	86	73	64
7	275	191	147	119•	100	87	76
8	296	213	166	136	115	100	88
9	316	233	185	153	131•	114	101
10	333	252	203	170	146	128•	114
11	349	271	221	187	162	143	127•
12	364	288	239	204	177	157	141

PB = 16 (Average DASD Path Busy %)

RT/ST = SKEW	1.30	1.35	1.40	1.45	1.50	1.55	1.60
3	429	154	94	67	53	43	37
4	446	190	121•	89	70	58	49
5	458	222	147	110	87	73	62
6	466	250	171	130•	105	88	75
7	471	275	194	150	122•	103	89
8	476	296	215	169	139	118•	103
9	479	316	235	188	156	134	117
10	482	333	255	206	173	149	131•
11	485	349	273	224	190	165	146
12	486	364	290	242	207	181	161

PB = 20 (Average DASD Path Busy %)

RT/ST = SKEW	1.30	1.35	1.40	1.45	1.50	1.55	1.60
3	n.a.	403	154	95	69	54	44
4	n.a.	426	190	123•	90	72	59
5	n.a.	441	222	149	112	89	74
6	n.a.	452	250	173	132•	107	90
7	n.a.	460	275	196	152	124•	105
8	n.a.	466	296	217	172	142	121•
9	n.a.	471	316	238	190	159	136
10	n.a.	475	333	257	209	176	152
11	n.a.	478	349	275	227	193	168
12	n.a.	481	364	292	245	210	184

Generalized LDR. Actuators required, traffic = 10000 ms/s. For other traffic levels, e.g. D.TRF, multiply the Table entry by D.TRF/10000

FIGURE 9 : Generalized LDR, Actuators

Group	1	2	3	4	5	6
observations	90	16	74	28	10	18
total hours	872	123	749	910	147	763
average CT	5.1	5.1	5.1	5.0	5.0	5.0
average SKEW	6.8	7.7	6.6	6.3	7.1	6.1
RT/ST						
average	1.32	1.46	1.29	1.33	1.37	1.31
minimum	1.03	1.22	1.03	1.11	1.21	1.11
maximum	1.63	1.63	1.58	1.48	1.53	1.48
% error, SKEW						
average	0.6	-2.7	1.3	1.1	1.0	1.2
absolute	5.7	6.3	5.5	4.3	4.8	4.3
% error, P-K						
average	6.4	0.3	7.7	5.9	3.2	7.3
absolute	9.4	5.9	10.2	7.7	6.2	8.5
ST						
average	19.4	18.6	19.7	19.3	18.7	19.5
minimum	14.3	16.4	14.3	14.9	16.6	14.9
maximum	24.9	22.3	24.9	23.2	22.3	23.2
% error						
average	0.2	-1.2	0.4	0.1	1.0	-0.4
absolute	7.4	6.2	7.7	6.5	5.6	6.9

FIGURE 10 : Validation

CICS	Customer Information Control System
CPU	Central Processing Unit
CU	Control Unit
DASD	Direct Access Storage Device
DLS	Device Level Selection
DPR	Dynamic Path Reconnect
DPS	Dynamic Path Selection
IMS	Information Management System
I/O	Input/Output
LCU	Logical Control Unit
LDR	Law of Diminishing Returns
MVS	Multiple Virtual Storages (operating system)
MVS/SP2	MVS Systems Product, Version 2 (XA)
MVS-XA	MVS Extended Architecture
RMF	Resource Measurement Facility
RT	Response Time (ms) of a DASD I/O
SCP	System Control Program
SIO	Start I/O
ST	Service Time (ms) of a DASD I/O
TSO	Time Sharing Option

FIGURE 11 : Acronym definitions

LARGE CAPACITY DASD: IS PERFORMANCE SOMETHING TO BE AFRAID OF?

Bruce McNutt
International Business Machines Corporation
San Jose, California

ABSTRACT

Technology has made possible a rapid increase in the density of DASD magnetic media. Performance analysts and capacity planners have been faced with a long and continuing succession of DASD products offering sharply larger storage capacity. With the larger capacity comes an enlarged scope of action for each actuator, and the concern that the actuator might have trouble keeping up. But so far, performance on new and larger DASD devices has repeatedly contradicted expectations by turning out to be a non-problem.

This paper investigates why it is that large-capacity devices keep turning in better-than-expected performance. The starting point of the paper is that our expectations tend to be guided by the view of a DASD device as the "server" in an M/M/1 queueing model. This view does not lead to correct intuitions about DASD performance, because it fails to take into account the task structure of I/O activity. The critical impact which task structure has on device queueing, on skew, and on arm movement is examined. By considering all of these areas collectively, it becomes clear why the task structure of I/O activity tends to favor devices with large capacity -- and why such devices tend to perform well even when we don't expect them to.

1. INTRODUCTION

Technology has made possible a rapid increase in the density of DASD magnetic media. Performance analysts and capacity planners have been faced with a long and continuing succession of DASD products offering sharply larger storage capacity. With the larger capacity comes an enlarged scope of action for each actuator, and the concern that the actuator might have trouble keeping up. But so far, performance on new and larger DASD devices has repeatedly contradicted expectations by turning out to be a non-problem.

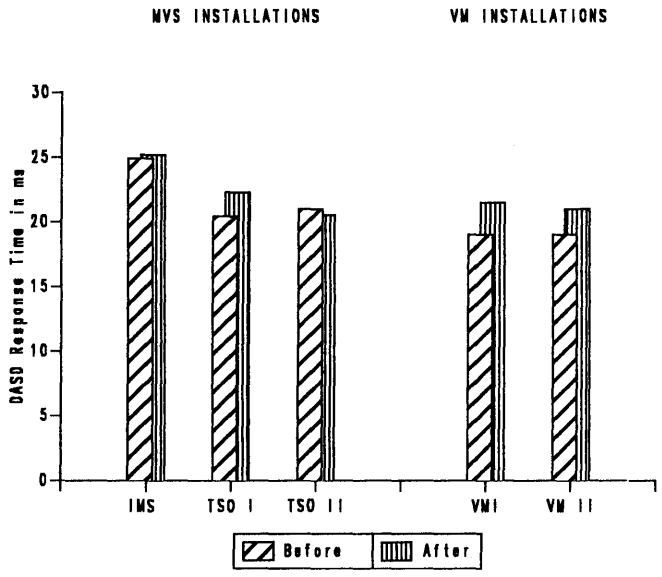


Figure 1. Results of folding at five installations: before and after

Following the introduction of the 3880 Model E DASD, which offered twice the storage capacity per arm as the previous 3880 Model A device, formal studies were done to determine the impact of the larger device size on performance [1, 2]. These studies confirmed that, as in previous cases when a device with larger capacity has been introduced, the larger capacity is not harmful in practice. Data was folded onto 3380 Model E devices at one IMS shop, and at two TSO and two VM installations. Figure 1 summarizes the overall DASD response times before and after folding as observed in these studies. At the IMS site, response time increased by roughly a third of a millisecond; at one TSO site, response time decreased by about half a millisecond; and at another TSO site, response time increased by about 2 milliseconds. The observed increase in average response time at the two VM sites was also about 2 milliseconds. At each site, the amount of data per arm was doubled or (in the IMS case) tripled; yet folding never resulted in a serious performance problem at any of the sites.

This paper investigates why it is that large-capacity devices perform so well in realistic installation environments. The starting point for the investigation is to understand better the fine-structure of the I/O activity which typically occurs on a DASD device.

In normal modeling of DASD, a very simple model of I/O fine structure is assumed. The M/M/1 technique for modelling device activity, for example, assumes that the distribution of I/O interarrival time is exponential and that each I/O is independent. The possibility of I/O "bursts" is not considered, nor is the fact that long strings of I/O's can be generated as the result of executing a single task.

Figures 2 and 3 illustrate just how far astray we can go if we accept these assumptions at face value. Figure 2 breaks down a specific sequence of I/O requests generated over a one-minute period according to the assumptions of the M/M/1 model. By contrast, Figure 3 breaks down the I/O pattern observed on a specific TSO user data volume over an identical period of time.

The average I/O rate in both figures is 2.9 I/O's per second. Each figure shows how this average rate breaks down into I/O rates observed during very short sub-intervals of time (lasting 0.1 seconds). The I/O rate in a given sub-interval is the number of I/O's which occurred during that sub-interval divided by the interval length of 0.1 seconds -- i.e., the I/O rate in a given sub-interval is ten times the number of observed I/O's. As Figure 2 shows, the M/M/1 requests are spread out more or less uniformly across the one-minute period. This uniform spreading contrasts sharply with the bursts of activity in the live data of Figure 3.

Although Figure 3 refers specifically to a TSO user data volume, bursts of activity of the type which the figure illustrates are also typical of IMS and CICS data base volumes and of VM user data volumes (a broader discussion of I/O bursts as seen on a large number of VM DASD appears in [3]). Note that despite the average I/O rate of 2.9 I/O's per second, the device is idle most of the time. It becomes active when some task initiates a rapid sequence of I/O's -- usually without interference from other tasks. Figure 3 shows that during such a burst of activity, the I/O rate may briefly reach 60 to 70 I/O's per second. Since the burst is initiated by a single task, however, it does not cause queueing.

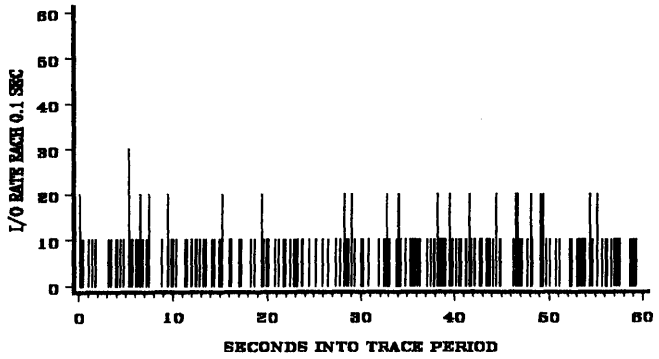


Figure 2. Poisson arrival I/O rates for each 0.1-second sub-interval of a one-minute period. Average I/O rate = 2.9

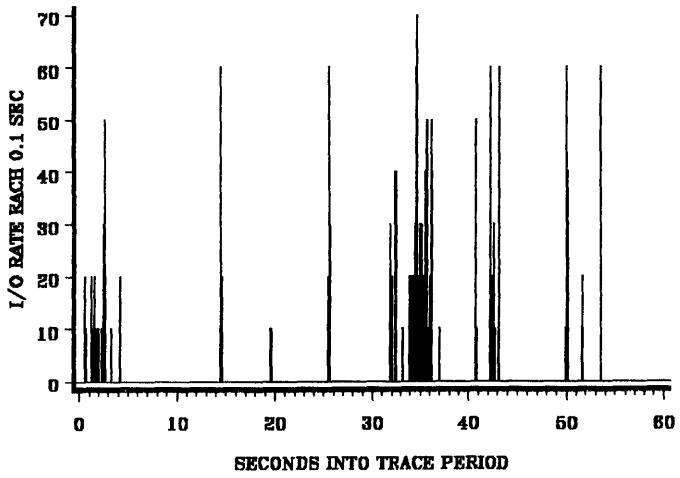


Figure 3. Traced TSO volume I/O rates for each 0.1-second sub-interval of a one-minute period. Average I/O rate = 2.9

As long as tasks do not interfere seriously with one another, there is every reason to use the largest available size of DASD device. The cost per megabyte of storage decreases with the size of the device; and at the same time, the actuator is able to handle the demands of any one task without much difficulty. Figure 3 makes clear that the opportunity to achieve *both* economic storage *and* good performance may be missed if the decision is guided by reasoning based on the M/M/1 model.

There is no claim here that large capacity devices are always the best choice for every type of data -- only that the choice of the most suitable device depends upon the fine structure of the I/O pattern. A pool of performance-critical system volumes containing data in use by large numbers of users -- e.g., master catalogues, heavily used program libraries, IMS long message queues, Resource Access Control Facility (RACF) data -- should not be folded 2-to-1 onto a larger capacity device. In this case, too many tasks may end up using a single actuator at the same time.

On the other hand, we do not need to be as cautious when it comes to data base and user data volumes. This is because active tasks are usually spread thin across such a pool. The thin spreading of tasks tilts the odds in favor of the devices of the pool being able to perform well even if their capacity is large. In particular, for both small-capacity as well as large-capacity devices containing data base and user data storage,

1. the small number of tasks using a device at any one time limits the degree of device queueing,
2. peak I/O rates are associated mainly with surges of single-task activity, and
3. seek times are dominated by the activities of individual tasks.

Since performance is mainly a function of individual task behavior, the performance provided by a larger-capacity device can rival that provided by a smaller device.

In the rest of the paper, we build a better understanding of the impact of I/O fine structure by investigating each of the phenomena (1 - 3) in turn. Some new data on device queueing are presented, and we also pull together the results of previous studies on skew [4] and on arm motion [5], with a focus on how the results of these studies bear on the performance of large capacity devices.

2. DEVICE QUEUEING

The M/M/1 model is so widely used, so powerful, and so easily applied that the temptation to think of a DASD device as the "server" in an M/M/1 queueing model is very hard to resist. This view may, in fact, closely approximate a few I/O devices on the system such as a paging device or the volume containing a heavily used master catalogue. Highly active system volumes of this kind, however, are not typically candidates for folding onto larger capacity devices. Usually, the impetus to use large capacity devices is caused by increasing demands for data base storage and storage of user data. On a typical data base or user data volume, the pattern of activity departs drastically from the assumptions of the M/M/1 model.

Figure 3 presents a good explanation of how this departure occurs on one specific device. But to see how dangerous an unqualified reliance on the M/M/1 model can really be, it is necessary to try applying it to all of the volumes at an installation. The result of doing this at nine large CICS, IMS, and TSO installations is presented in Figure 4.

The data analyzed in Figure 4 were obtained as the result of an ongoing project to collect live DASD workload data. Other results of this project and additional details about the collection of the data are given in [4]. Figure 4 is based on examining a single peak-period Resource Measurement Facility (RMF) interval at each of the nine MVS/XA installations at which data were collected. Altogether, 3255 volumes were observed at the nine installations. For each volume, the RMF queue length data was examined. The average queue length is a measure of total queue activity; it is equal to the product of queue time and I/O rate. In other words, the average queue length is the sum of the time per second spent on the device queue by each task.

Figure 4 shows a detailed analysis of the 53 volumes of the study which had a sufficiently high utilization, during the period being examined, so that the M/M/1 model would predict a queue length of at least 0.2[1]. The RMF data for each volume of Figure 4 are shown as a single point. The average queue length observed on the volume, plotted on the vertical axis, is compared to the M/M/1 estimate of the average queue length on the horizontal axis. The M/M/1 estimate is computed as

$$L = \frac{\rho^2}{1-\rho} \qquad (1)$$

where ρ is the utilization of the device as reported by RMF.

The various volumes shown in Figure 4 appear to fall into two categories:

[1] Volumes on which active multi-host sharing caused extreme amounts of pend time are not included, however. The contention mechanisms, such as reserve/release, which arise in the case of shared DASD are different from ordinary device queueing of the type being considered in Figure 4, and in practice their effects can be more severe. Also, it is not clear how to take into account extreme amounts of device pend time in applying the M/M/1 model.

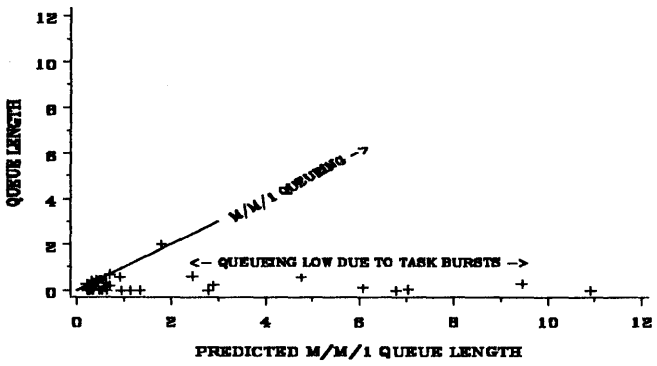


Figure 4. Tests of the M/M/1 model: predicted and actual queue length

1. volumes for which the M/M/1 model works, which lie along the 45 degree line extending from the origin, and
2. volumes for which the queueing is much less than that predicted by the M/M/1 model, which lie in a flat band along the bottom.

Volumes of type (2) can sustain very high loads without paying much of a penalty in queueing time. Utilization levels for these volumes occasionally reached 80 or even 90 percent. For a utilization of 80 percent, the M/M/1 model would predict a queue length of 3.2; yet the queue length along the flat band never exceeds about 0.6.

On the other hand, load levels of 80 percent on a volume with M/M/1 behavior can quickly result in loud complaints about response time. This helps to explain why no load levels this high were observed for the volumes of type (1).

Fortunately, only a minority of volumes fall into category (1). If we define these volumes formally by using the criterion that the actual queue length must be no less than 90 percent of the M/M/1 prediction, then they account for only 19 percent of the volumes presented by the figure, or 15 percent of all active volumes included in the study. The M/M/1 model works for these volumes because they are continually being shared by a large number of users.

By contrast with the minority of volumes in category (1), most data base and user volumes are *not* being shared by a large number of users at any given time. To put this into better focus, Figures 5 - 7 show a detailed breakdown of contention for the devices in three specific pools of user data, each belonging to a specific installation participating in the study described earlier. A trace of the I/O activity in each pool was used to prepare the three figures. The period of the trace was divided into very short sub-intervals (lasting 0.1 seconds) for the purpose of determining how many tasks were active at a given moment in time. A task was considered to be "active" on a specific device during a given sub-interval if the task produced at least one I/O to the device during this period of time.

Figures 5 - 7 present the frequency with which no tasks, one task, two tasks, or three or more tasks were observed to be active on a given device based on the three sets of I/O traces. Figure 5 presents data for an IMS data base pool of forty-six 3380 Standard volumes with an average I/O rate of 4.4 I/O's per second; Figure 6 presents data for a TSO user data pool of sixty-seven 3380 Standard volumes with an average I/O rate of 2.2 I/O's per second; and Figure 7 presents data for a VM user mini-disk pool of twenty-eight 3350 volumes with an average I/O rate of 3.1 I/O's per second. For each of these pools, the devices were idle (not being used by any task) most of the time. If a device was not idle, then it was probably being monopolized by a single task; and if it was not being monopolized by a single task, then it was probably being used by exactly two tasks. Thus, most queueing involved two tasks which took turns on the device.

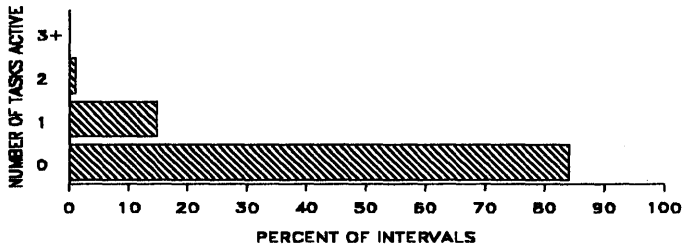


Figure 5. Histogram of the number of active tasks: 46 IMS data base volumes

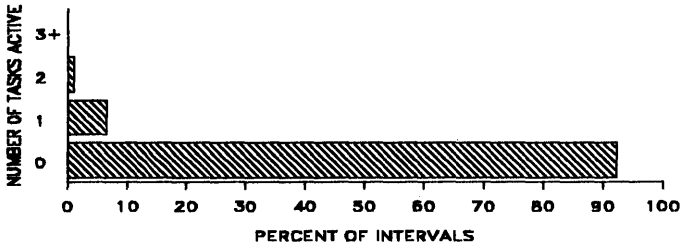


Figure 6. Histogram of the number of active tasks: 67 TSO user data volumes

It is interesting to compare the M/M/1 prediction of queue length given by (1) with the behavior of a queue in which two tasks take turns in using the device. In the latter case, the queue length can never exceed unity no matter how heavily the device is utilized. But in the M/M/1 case, it is possible for the predicted queue length to become arbitrarily large, depending on the value of the utilization ρ. The comparison becomes even more dramatic if the device is monopolized by a single task, since in this case it is not even possible for queueing due to contention between tasks to occur[2]. Since the cases of single-task monopolization and two-task sharing are the most common cases of device use, it is not hard to understand the dramatic failure of the M/M/1 queueing model as shown in Figure 4.

It is interesting to ask whether queueing would continue to be so well-behaved if the pools of data represented in Figures 5 - 7 were folded 2-to-1 onto larger devices (the 3350's onto 3380 Model D's and the 3380 Standards

[2] Since some queueing can occur even in the absence of contention between tasks, the queue length may not be precisely zero in the case of single-task monopolization. For example, both QSAM and BSAM are capable, under some circumstances, of "stacking" more than one request from the same task onto the queue for the same device.

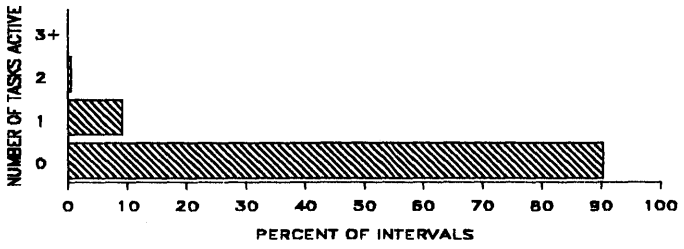


Figure 7. Histogram of the number of active tasks: 28 VM user mini-disk volumes

onto 3380 Model E's). This question can be answered by referring again to the same data used in preparing the three figures.

Suppose that in folding one of the pools presented in Figures 5 - 7, each folded pair of volumes is chosen at random from the original volumes of the pool. Then the probability that a given folded volume has a given number of tasks active in each sub-interval can be calculated combinatorially. For example, the number of volume pairs with two or more tasks active is the number of pairs in which both members of the pair have at least one task active, plus the number of pairs in which one member is idle and the other member has two or more tasks active. When this calculation is applied to the IMS data of Figure 5, the probability of two or more tasks being active increases from .0097 in the non-folded case to .0416 after folding. For the TSO data of Figure 6, the probability increases from .0118 to .0280; and for the VM data of Figure 7, the probability increases from .0057 to .0197. In each case, the probability increases but remains *very* small. Queuing would continue to have little impact on the performance of these pools of data.

It should be noted that a number of models of DASD activity more sophisticated than M/M/1 are currently available in the form of packaged analysis tools [6, 7]. Models capable of describing the CPU and associated DASD as a closed system are of particular interest. Although these models do not usually account in detail for the number of tasks using each device, they do account for the fact that a specific finite number of tasks (the multiprogramming level) are running in the system at any given time. This limits the total amount of queueing which can occur across all of the devices, and greatly improves the realism of the results.

3. SKEW

I/O rate "skew" across the volumes of an installation has a critical impact on the effectiveness of large-capacity DASD, because skew dictates the volume loads which will result from folding larger amounts of data onto each device. Two contrasting extremes help to illustrate this:

1. No skew: I/O activity is divided uniformly across all devices. In this case, 2-to-1 folding of data onto a larger capacity device doubles the load on every device.
2. Maximum skew: all I/O activity is concentrated on a single device. In this case, 2-to-1 folding of data onto a larger capacity device does not cause any change to the load on the single device which is being used.

Obviously, realistic skews exist somewhere between these two extremes. In most installations, however, the distribution of the I/O across volumes is *very* uneven. As a result, the impact of 2-to-1 folding is described better by (2) than by (1).

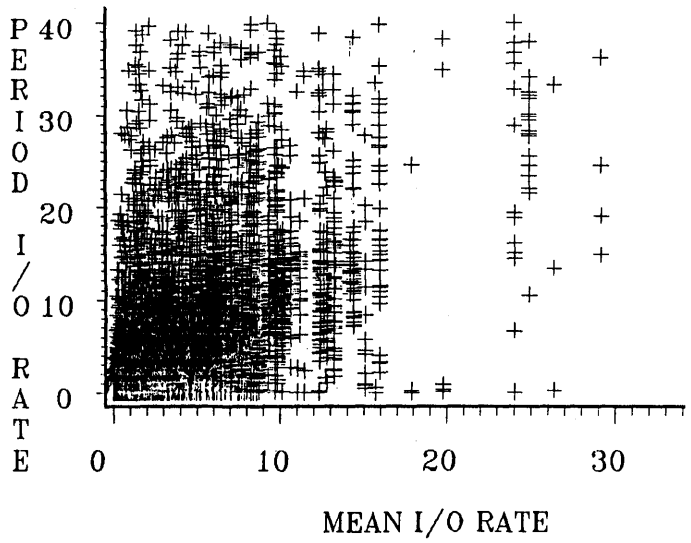


Figure 8. Variability of I/O rates on data base and user data volumes

Sharp skews are the norm because a typical installation supports a wide variety of activities. The amount of I/O generated by a given task running on the CPU is highly unpredictable, as is the specific set of DASD to which the task's I/O is directed. Figure 8 illustrates the chaos which results. This figure is taken from [4]. It shows the I/O rates which were observed on 2879 data base and user data volumes, at ten installations, during a day's worth of peak-load RMF intervals.

Each point of Figure 8 corresponds to a single volume observed during a single RMF interval. The I/O rate of the volume during this specific interval is shown along the vertical axis, and the average I/O rate of the volume over the entire day is shown along the horizontal axis. As the figure makes clear, the activity of a given data base or user data volume during a given RMF interval is extremely unpredictable.

Although it is difficult to predict the behavior of a single volume, however, we can certainly attempt to predict the statistical behavior of a group, or pool, of data base or user data volumes taken collectively. Each volume of the pool can be viewed as an observation of a random variable. This random variable, in turn, is the sum of other random variables associated with the activities of individual tasks. In [4], this approach is applied to obtain statistical predictions of skew, based on data observed at the ten installations described above.

	PREDICTED HIGHEST I/O RATE WHEN N IS ...						
	8	16	32	64	128	256	512
A							
1	3.8	6.1	9.8	14.7	21.7	29.4	36.9
2	6.8	10.5	15.8	22.6	30.6	38.0	44.2
4	12.1	17.7	24.6	32.7	40.2	46.7	57.6
8	21.2	28.6	36.9	44.6	52.3	62.6	71.6

Figure 9. Predicted I/O rate of the busiest of N volumes, given an average volume I/O rate of A

To understand the effect of folding on skew, it is useful to examine the I/O rate of the most heavily used volume. Figure 9 presents the statistical predictions of the I/O rate to the busiest device which are developed in in [4]. The figure shows how to project the I/O rate for the busiest of N active volumes, assuming that the average I/O rate across the N volumes is A.

The result of 2-to-1 folding is to increase A by a factor of two and to decrease N by a factor of two. Figure 9 shows that, in a statistical sense, the

combination of these two changes causes only a very slight increase in the load being placed on the busiest volume. For example, suppose that before folding a pool of 32 user volumes has an average I/O rate of 2 I/O's per second. Then before folding, the projected I/O rate to the busiest volume is 15.8 I/O's per second. After folding, the projected I/O rate to the busiest volume is 17.7 I/O's per second. By examining this case and a variety of others like it, it is easy to verify from the figure that folding is unlikely, *by itself*, to introduce new performance problems if the per-device loads are light enough so that folding makes sense.

Again, the claim is not that performance problems don't happen -- only that they are unlikely to be caused by taking a pool of data base or user data volumes which has reasonably light average utilization and folding this pool 2-to-1. This point is made clearer by reflecting for a moment on how DASD performance problems actually *do* occur. When the phone rings and fire fighting is needed, what does a performance analyst look for first?

The first suspicion is usually that there is some specific data set performing a system-wide service which is receiving more use than it can handle. Examples would be master catalogs, global indexes, load libraries, paging data sets, RACF data sets, and so forth. If a data set of this kind is used too heavily *by too many different users and tasks*, then a performance problem will result. This will happen regardless of whether small or large capacity devices are used, since the data set has to go *somewhere*.

It may also happen, of course, that two or more data sets which are used heavily by many users are placed on the same volume. This may even be done on purpose, if the data sets are logically related to each other -- not by the performance staff of course! If the pool of *system* volumes is folded 2-to-1, then folding may create performance problems due to this effect. This is the reason that 2-to-1 folding is safer on data base and user data volumes than it is on system volumes. Nevertheless, it is interesting to note that the same problem also occurs on small-capacity devices, and can be fixed equally well on either a small- or large- capacity device by separating the highly active data sets onto different volumes. In most cases, the problem can also be fixed (and a considerable performance benefit achieved as well) by placing highly active system data sets of this kind under a cached controller.

4. ARM MOTION

Arm motion is another area where the task structure of I/O is important. The tendency for single tasks to dominate the use of a given device has a highly beneficial impact on the seek times of any sort of DASD, regardless of size. We review the effect in this paper because a large-capacity device still sees most of this benefit, despite its size.

The impact which the task structure of I/O has on typical device seek times is dramatic. For example, the seek time required for a seek between two random locations on a 3380 Model D device (the so-called "nominal" seek time) is 15 milliseconds. On a 3380 Model E device the nominal seek time is 17 milliseconds. Yet, true average seek times on 3380 Model D and Model E devices operating in live environments are typically in the range of 2 to 7 milliseconds [5].

Real average seek times are so much smaller that "nominal" seek times because if the data set being requested by a given I/O was also requested by the previous I/O, then this data set is already under the arm. The arm may not need to move; or if movement is needed, the seek distance is likely to be short. Taking into account both cases of no arm movement as well as the distribution of seek distances if arm movement is necessary, the average seek time for an I/O request to a data set which is already under the arm is no more than two or three milliseconds for a device type in the 3380 family. Only occasionally is it necessary to reposition the arm to an entirely new data set, thus requiring a "nominal" seek.

This same pattern of behavior occurs on both larger and smaller capacity devices. The relationship between the two cases can be explored by asking what happens when data is folded 2-to-1 from a smaller device onto a larger one.

Before folding, each task that is active on a given device causes the arm to move according to a pattern required by the task itself. This pattern may involve both seeks within the same data set as well as "nominal" seeks between different data sets. The pattern is occasionally interrupted by the activity of some other task on the device, but as Figures 5 - 7 suggest, the level of interference is minimal. The overall seek time is dominated by the seek activities of individual tasks.

As the result of folding, the level of interference between tasks must increase, and hence some new "nominal" seeks must occur. But how significant is this added arm motion compared to the arm motion associated with the activity of the various individual tasks of the system, absent any interference with each other?

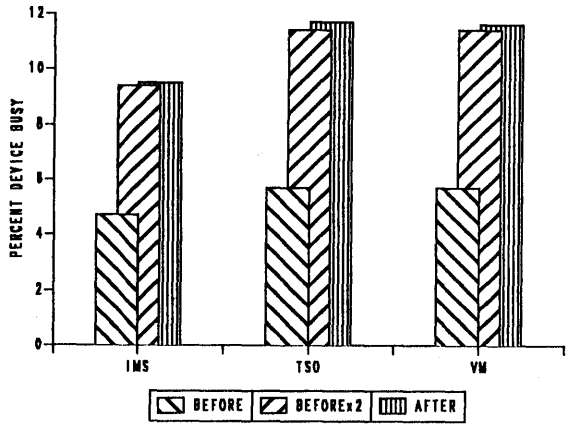


Figure 10. Impact of 2-to-1 folding on device utilization for three pools of data

A detailed examination of this issue is performed in [5]. Using GTF and VM Monitor seek traces, the seek times before and after folding are computed for several pools of data, including IMS data base storage, TSO user data, and VM mini-disk storage. The pairing of volumes is taken to be random; pairs of volumes whose I/O patterns take turns are taken into account, as well as pairs whose I/O patterns conflict.

A summary of the impact of folding on device utilization in the TSO, IMS, and VM case studies of [5] is presented by Figure 10. For each pool, the figure shows the utilization per volume prior to folding and after folding. For comparison, the figure also shows the utilization per volume which folding would have produced assuming no change in seek times (exactly twice the original utilization per volume). As the figure makes clear, the added utilization caused by additional seek time is insignificant. The arm motion required by the activities of single tasks predominates over any arm motion caused by contention between tasks, and this is true both before and after folding[3].

5. CONCLUSION

As we have seen, active tasks tend to be spread thin across data base and user data volumes. These volumes spend most of their time either idle or being monopolized by a single task. Long periods of inactivity alternate with short bursts of high demand. Since the high demand is usually created by one task at a time, such demand does not create additional queueing or lengthen device seek times. Thus, these volumes are likely to perform well even if they are used to store large amounts of data. This has encouraged the introduction of a long and continuing succession of DASD products offering sharply higher capacity per volume.

[3] [5] puts these statements into a more quantitative framework. Using a combinatorial technique similar to that of Section 2 of this paper, the mild impact of 2-to-1 folding on seek time can be shown to be the direct result of the fact that "accidental" arm steals between data sets, as caused by task contention, are unimportant.

Meanwhile, the cost advantages of increasingly large storage media have not been lost on the developers responsible for new data base applications. It has been possible to cost-justify new applications with increasingly large storage demands. On the average, such applications tend to further thin out the density of tasks with respect to the data on which the tasks are operating; and this in turn further encourages the introduction of new and larger devices.

This situation has continued to evolve ever since the earliest DASD. In 1972, W. L. Lynch reported measurements of 2314 devices running under MTS (Michigan Terminal System) which showed surprisingly little arm movement [8]. The patterns of access to data base and user data volumes observed in this paper have persisted despite the more than 40-fold increase in device size between the 2314 and the 3380 Model E.

Clearly, then, these patterns do not depend on our having examined some specific generation of devices and applications. Instead, they would appear to be a stable characteristic of data base and user data volumes.

We have tested this stability further by surveying the impact of 2-to-1 folding on queueing, skew, and arm motion as observed in several test cases involving live installation data (some introduced here, others in [4] and [5]). This is a particularly stringent test, because 2-to-1 folding does not expand the data used by a given task or cause any thinning of the tasks with respect to the data on which they operate. Even so, the predominance of activity by single tasks as observed in all of the test cases surveyed continues to persist even if the data for each test case is folded 2-to-1.

It seems reasonable to conclude that the phenomenon of large devices which perform surprisingly well is likely to be be with us for some time. It is not necessary, however, for us to continue being surprised by it.

REFERENCES

1. J. R. Garner and R. A. Lindsay, "IBM 3380 Extended Capability DASD Early Experiences," Washington Systems Center Bulletin GG66-0226, October 1985.

2. Bob Campbell, "SJEVM 3380E Project," presentation to SHARE, August 1986.

3. B. McNutt, "A Case Study of Access to VM Disk Volumes," CMG XV Proceedings pp. 175-180, December 1984.

4. B. McNutt, "An Empirical Study of Variations in DASD Volume Activity," CMG '86 Proceedings pp. 274-283, December 1986.

5. B. McNutt, "The Impact of 2-to-1 Volume Folding on Seek Time: A Comparison of Five Cases," CMG '85 Proceedings pp. 427-433, December 1985.

6. BGS Systems Inc., "Best/1-MVS Reference Guide," Doc. No. 85087-00, 1986.

7. Boole & Babbage, Inc., "CMF User's Guide," Doc. No. TD2004-3, August 1986.

8. W. L. Lynch, "Do Disk Arms Move?", ACM SIGMETRICS Newsletter, v. 1, pp. 3-16, December 1972.

CAPACITY PLANNING FOR A MULTI-PROCESSOR ENVIRONMENT SUPPORTING CICS TRANSACTIONS

Donald T. Gantz *, Carlos M. Gonzalez *, and Melvin L. Moore **
* George Mason University, ** U.S Customs Service

ABSTRACT

In this paper we discuss an ongoing capacity planning effort at the U.S. Customs Service which has a large and growing online data processing requirement. Capacity planning is implemented for IBM 3084 and 3090 based systems running large CICS workloads under the MVS/XA operating system. Measurements on SMF and RMF data are made using as a tool Merrill's MXG software. Performance data bases are designed for efficient information retrieval and for maintenance of an organized historical record of the data. Resource delivery and performance of the system are profiled and reported using programs written in SAS. Innovative graphic summaries are used to provide succinct, informative management reports. Reporting includes status snapshots, trend analysis and projections. Data collecting and reporting are linked to a menu driven system for job submission to facilitate scheduled reporting carried out by quality assurance personnel. We discuss some of the changes that have been made as a result of this capacity planning effort.

1. INTRODUCTION

Most of the work done in capacity planning [BRO-77; BRO-79; and COR-83] has used service resource units as a measure of capacity for the system. This measurement technique addresses the question of how resources are being used and it has a potential for forecasting available capacity for future needs. Additionally, an analysis done for CICS systems [FRI-86; PAT-85; and WON-86] requires measurement of the response time of the system. Response time measurement addresses the question of how well service is being provided to users. One of the purposes of this work is to report our experiences with a management model which integrates data on resource utilization and quality of service to users. An important objective in designing the model is the use of graphics as a means to present results to management. Techniques of exploratory data analysis [CHA-83; MOS-77; HOA-83; and TUK-77] are used to obtain the desired graphical reports.

This paper is subdivided into sections which explain the working environment, the measurement tools utilized, the measurement techniques, an analysis of the results, a management perspective, and a final section of conclusions and future research.

2. WORKING ENVIRONMENT AND MEASUREMENT TOOLS

The work reported here was done for the U.S. Customs Service for their computer center located in Alexandria, Virginia. When the project started, the center had an IBM-3084 running the MVS/XA operating system and servicing more than 2300 terminals scattered throughout the U.S. The volume of work generated by these terminals was on the order of 250,000 CICS transactions per day. As the number of applications grew, the center added an IBM-3090/200 also running under MVS/XA; This machine was recently upgraded to a 3090/400. The number of CICS transactions is currently more than 460,000 per prime shift, with still more applications to be implemented. Most of the CICS load is handled today by the 3090, leaving the 3084 for all the testing and non-CICS applications.

The data center utilizes a variety of monitor and report generator tools to assist in tuning and managing the systems. In particular, OMEGAMON is used as a monitor tool and reports are generated using SLR and JARS. Reports are generated and used as needed and then discarded. There had essentially been no historic performance data base and no ongoing capacity assessment or modeling. This paper therefore discusses the first stages of setting a capacity planning capability rather than reporting on a new phase of an ongoing effort such as is reported in [KRA-86]. The initial task was to start maintaining a performance data base and to set up an ongoing management reporting process. All major SMF [SMF-87] and RMF [RMF-87] records are being journaled, and there is a considerable library of historic raw data. We decided to use SMF and RMF data for our analysis. Having chosen the accounting data as the primary data source, the next step was to choose a software system to decode this collection of records into a usable performance data base. The data in the accounting files are compatible but disjoint, with information spread across a number of different records in many sequences. Hence, we needed a good tool to facilitate this analysis. SAS has become a widely accepted standard for reading SMF data through its utilization by Merrill's MXG [MER-84] software and Morino's MICS [MOR-85] products. Merrill's MXG software is an inexpensive tool for reading journaled data and forming a performance data base of SAS data sets. Other products like MICS are more comprehensive tools for the capacity planner, but are considerably more expensive. With our background and experience with SAS and budget constraints, Merrill's MXG seemed ideal for our purposes. The MXG software provides us with an automated system that uses SMF and RMF data to build a performance data base (PDB). The PDB provides measures of resource consumption and response times, and it is our basis for analysis, reporting and planning. Merrill's MXG and SAS [SAS-85] were installed by the data center.

We set about the task of measuring and reporting what the computer is delivering and how well it is performing under its load. Our reports are to assist managers who are responsible for procurement and planning, and our measurements and reports have to be directed at these responsibilities.

Higher management are sensitive to a declared middle management separation of capacity planning and data center operations responsibilities. As a result, specification of control performance groups by the capacity planners might have appeared to constitute interference with performance tuning and data center operations. Control and report performance groups had been defined to meet the needs of computer systems operations personnel. The priority and origins of our work were not central enough to allow us to specify our needs for redefinition of the groups. We began our work constrained to live with existing definitions. As a consequence our ability to relate resource usage to the business units consuming them was affected.

Additionally, there is no formal chargeback system for data center services, and, consequently, user and job identifiers are not regimented or comprehensive. There is an effort underway to systematize user and job identifiers.

Our working relationship with the data center systems people is good, and we have accumulated the knowledge required to credibly execute our measurement and

reporting. Since Merrill's MXG is essentially a data extraction tool, we have to write our own SAS programs to manicure and analyze the extracted data and to report to management. We are well prepared for this requirement. One of the authors is an applied statistician with many years of experience in using SAS for data analysis and reporting. Further, much of this author's consulting and academic work has focused on innovative use of graphics for analysis and reporting. We have available many graphical techniques and the SAS routines to execute them.

The programming task included preparation of routines to properly clean and assemble the measured data. For instance, program statements were prepared that selected out real CICS transactions from SMF Type 110 records and purged routing and other extraneous transactions. This sort of analysis bookkeeping is particularly important to our reporting responsibilities. Management's concept of processed CICS transactions consists of the production workload that is the main business of the data center. It is important that non-business transactions generated by the system or by testing, etc., not inadvertently distort the accounting of business transactions. Additionally, it is vital for our reported work volumes to be compatible with volumes from other reports generated directly by the data center. Therefore, we not only have to be thoroughly aware of what we are actually measuring, but we have to know what is actually being measured by others.

3. MEASUREMENT TECHNIQUES

Initially data was analyzed for the whole 24 hour period in order to comprehend the nature of the business handled by the data center. After a preliminary analysis, we concluded that most of the work load was handled during the prime business working hours between 8:00AM and 4:00PM. We expanded our measurements and analysis to the period between 8:00AM and 7:00PM to include fully the west coast working hours.

We designed and implemented the procedures to create the PDB's necessary to maintain an efficient information retrieval system. At any given time we have the PDB's for :

* Daily -- for all days before the previous sunday.
* Weekly -- for all weeks of the current month, except the current week.
* Monthly -- for all months of the current year, except the current month.
* Yearly -- for all years, except the current year.
* The accumulated data for the current week.
* The accumulated data for the current month.
* The accumulated data for the current year.

Based on management's need to quickly access the information in the PDB's, we decided to keep on line only the PDB's of daily data for all the days of the current week, weekly data for last week, and the accumulated data for the current week. The rest of the PDB's are kept on tape, and updated when required.

We went through an iterative process of management review of report designs. After a comprehensive menu of reports was approved by management, we mapped back to our original PDB the variables needed to produce such reports. We then modified the MXG software for the creation of PDB's tailored to our specific needs. Thus, we left on the PDB only those variables needed to produce the required reports. This process eliminated the need for large disk storage, as well as increased speed of the information retrieval.

We created tapes with SAS data sets of SMF Type 110 records. The runs of our modified MXG programs could create data sets from the SMF/RMF data with relatively little CPU time. Below is a list of the SAS Data Sets built by the modified MXG software to create the PDB's.

Data Set	Type Title	SMF Record
TYPE70	RMF CPU activity	70
TYPE71	RMF Paging and Swapping activity	71
TYPE72	RMF Workload activity	72
TYPE73	MVS/XA Physical Channel activity	73
CICSTRAN	CICS Transaction data	110

Table 1

4. DATA ANALYSIS AND MANAGEMENT REPORTS

One of the motivations for writing this paper is to make known to other technicians some of the quantitative tools that have gone over well with management in the current work. To this end, some examples of management reports are provided. These reports either summarize a single day's processing or attempt to compare daily processing over some period of time. These reports are produced using some SAS display procedures which produce a particular type of graph, but allow specifications on style and formats. Considerable use is made of SAS's capabilities for data manipulation and summarization, general purpose programming, and report writing.

We will first discuss our reports which summarize a single day's processing. Figure 1 is a CPU busy profile for one day's prime time processing. Management can see the peak and valley pattern (persistent from day to day), the available CPU time for growth, and the variation of the pattern across CPU's. This chart was a simple product of 'PROC CHART' in SAS. Additional reports in a generally similar chart style are presented in Figures 2 and 3. Figure 2 summarizes a day's delivery of resources. Resources are measured in 'resource units' and displayed in separate charts for CICS, database and other uses. Again management can see the pattern of delivery and compare consumption for the separate uses. The information in Figure 2 is from Type 72 RMF records which measure system resource usage. Type 72 records measure CPU time delivered to specific performance groups while Type 70 records measure all CPU time consumed. Figure 3 presents information from both Type 70 and Type 72 RMF records. By combining this information, the chart in Figure 3 includes the additional category of 'overhead' CPU time. The information in these three charts is basic and of clear management interest. The same format of these charts is used to present busy patterns for channels (from Type 73 records), page rates (from Type 71 records), or other RMF data information.

The measure of how well the system is servicing its users is the response times for CICS transactions. The CICS response time measured on Type 110 records is response time within the computer, that is, internal response time. A report on response times should give information concerning the system's success at meeting service level agreements. Typically such agreements reference the fraction of users who will experience a specified interval of response times. Figure 4 presents a statistical graphic which provides a succinct picture of response times experienced by users. The statistical graphic is the Box and Whisker diagram [TUK-77]. Box and whisker diagrams summarize percentile values of a measurement. The diagram consists of a box that runs from the 25th to the 75th percentile values, and whiskers extending from the box to extreme percentiles. In the diagrams the lower whisker indicates the 10th percentile and the upper whisker indicates the 95th

percentile. These selections reflect the relative importance of outlying values in either direction. The median (50th percentile) and the mean (average) values are indicated by using the symbols 'M' and 'A', respectively. Figure 4 presents a summary of one day's prime shift CICS response times using Box and Whisker diagrams. Note that the actual number and percent of CICS transactions in each hour is also included. This report immediately tells management the fraction of users experiencing various response times, and also tells a manager how response times vary over the shift. Management has readily accepted this style of report. The report conveys clearly and crisply much more information than tables of percentiles or charts of a single percentile such as the median or the 95th percentile. In this case there are no formal service level agreements; therefore, a traditional chart indicating the percentage of users for whom an agreement is met is inappropriate.

Additional reports compare daily processing over some period of time. Each of the reports in Figures 1 through 4 has a corresponding report style combining information from a number of days. Bar charts, such as those in Figures 1 through 3, for a number days can be aggregated to a single chart that summarizes the heights of the bars for all days. When there are just a few days, or a week, the span of heights can be indicated symbolically by |----------|; if there is a reasonably large number of days, then the variation in heights can be indicated by a Box and Whisker diagram. The Box and Whisker option is used in Figure 5 which summarizes system resource units delivered for a five month period. Figure 6 breaks down the 2PM data in Figure 5 into one month cuts. The graphs in Figures 5 and 6 are each produced through a trivial modification of a single program that runs on a continually updated summary data set of Type 70 and 72 RMF records. The structure of programs and data bases allows prompt response to management's desires for particular comparisons.

The preceding paragraphs have shown separate reports on resource delivery (work done) and response times (quality of the work done). Figure 7 presents information concerning resource delivery and response times in a single report. The left half of the report tells how many system resource units were delivered at 2PM on that day while the right hand side tells how many CICS business transactions were serviced and what response times were experienced. This report is valuable to a manager in that improved or degraded performance should be visible in the report. Further, whether the change in performance is associated with a change in workload should also be reflected in the chart. Note that the bar representing system resource delivery is composed of C's, D's and O's, indicating how the resources break down by CICS, data base and other performance groups. The staff member desiring this style of graph can select from a menu allowing resource delivery to be represented by aggregate system resource units (as in Figure 7), CPU units, memory units or I/O units. Also, menu selection governs for what dates data will be displayed. Figure 7 shows a degradation in response times in late February due to certain CICS transactions becoming more complex. Also shown is the change in resource delivery due to a hardware upgrade (3090-200 to 3090-400) in mid April; here an increase in memory units accounts for the increased system resource units delivered. After the upgrade, 95th percentile response times, represented by the upper whisker, were reduced by about two seconds.

The above discussion presented the style of graphic information that our measurement effort is reporting to management. These reporting formats were worked out through the interaction of manager and technician in the attempt to get a better understanding of service delivery of the computer system. The ongoing effort is trying to extract information providing a more accurate understanding of the potential for growth in the system. The ultimate goal of the project is provision of meaningful support for acquisition planning.

To this point, our discussion has focused on reports that consisted of crisp graphic information that was readily digestible by literate managers. Now our work turns to conclusions which utilize some theoretical statistical tools. Here, when we come up with a number, we do not appeal primarily to a picture for credibility, but we also need to credit the 'black box' of statistical methods, in particular, regression analysis. From the very first conversation with management, we are asked to predict when the system will hit capacity. We will discuss some of our analyses which address that prediction.

For the first two months of operation, the 3090/400 has been running with an overhead of 18 to 20 percent of expended CPU units. Besides the CICS transactions accessing the data bases, the other workload is relatively small and controllable during the prime shift. Since the CPU units due to CICS and to the data base system are really measuring the same work intensity, both of these quantities should not appear simultaneously as independent variables in a multiple regression. If they did appear simultaneously, the sum of their regression coefficients would be a robust quantity, but the separate regression coefficients would be inconsistent. A surrogate variable for this combined CICS/data base workload is the number of CICS transactions processed. We demonstrate our application of regression analysis from that approach.

First, we subtract from the total expended CPU units the portion due to the non CICS workload. The remaining total business CPU units are then related to the number of transactions processed. The analysis is done on the busy period from 10AM to 4PM during the two months since the upgrade to the 3090/400. A scatter plot of the data showing this relationship is presented in Figure 8. This scatter plot has been cleaned of the obvious bad data and spurious outliers. Examination of the daily reports presented in Figures 1 through 4 is one step in that cleaning. A regression fit directly to the data in Figure 8 is unsatisfactory. Viewing the regression residuals will quickly indicate a bad fit to the data. However, if we focus on the data where the number of transactions is at least 40,000, a straightforward regression analysis provides an acceptable fit. The residuals for this regression fit are presented in Figure 9; note that there is no pattern in the residuals that indicates a real lack of fit. The fitted regression line is (for one hour):

CPU UNITS = 14,761,986 + 409 * CICS TRANSACTIONS.

The desire, of course, is to calculate how many additional CICS transactions will bring the system to the theoretical limit of CPU units. There are some obvious objections to this type of projection. First, we would be using our regression relationship to predict outside of the observed range of the number of CICS transactions. Also, the system has other limitations which will degrade response times before these projections are fulfilled. However, current work is proceeding using these techniques and others. As our system comes closer to having saturated periods, we will gain experience and determine which techniques work or work best.

Online Report Selection

An online menu driven system was designed and implemented. The objective of this system is for the selection of specific reports and time periods. The system runs under ROSCOE [ROS-84] using the ROSCOE Programming Facility (RPF) [RFP-84].

The system builds, verifies and submits the proper jobs for the production of the reports selected by the user from the different menus. An example of some of the menus can be seen in Figures 10 to 13.

5. MANAGEMENT VIEWPOINT

The intent of directing an effort to create a capacity planning capability in the Customs Service was to be in a position to know when we needed to proceed with procurement actions for additional computer resources. The agency now has two data centers with about 4500 terminals and micro and mini computers located in the field offices. One center has an IBM 3084 and an IBM 3090-400 with about 150 gigabytes of on-line storage. The second center now has a NAS XL 80 with 190 gigabytes of on-line storage. Additional processors of these sizes are forseen to be needed in the next few years. The data centers are connected to field offices through an X.25 packet switching network and an extensive leased line system of communications. Major new applications to support the law enforcement and commercial systems of the agency are being developed. Systems hardware procurement schedules are rather lengthy in federal agencies. Long range planning is essential if we are not to be run over by normal growth that occurs in an organization with a rapidly expanding mission, like that at the Customs Service.

Within the Customs Service there is an Operations Division with responsibility to perform the normal systems tuning effort. While the information produced by this project can be used to advantage in this effort there was no intent to provide all the systems tuning data that might be required to perform that task. Systems tuning data that might be obtained was seen as a nice by-product that could be given to the system software staff for their use. There is an organizational issue here. Each different staff has its own sense of responsibility for using capacity information. The procurement staff for the data center are not integrated with computer systems tuning personnel, each being in different sub-organizations of the agency. Therefore, conflicts of interest can arise. They should simply not be allowed to remain. It is a responsibility of anyone who wants to see himself or herself as a manager to look past conflicts to see the common good of accurate performance data. Someone may see the presence of information as a challenge to how well they do their job, and not want to have performance information available to anyone outside their own group.

Within Customs there were some basic agency management issues to be resolved that became goals of the project. These were as follows:

Procurement Planning. We want to know enough about how our workload is growing to be able to plan for acquisitions through the competitive process. We recently completed a large competitive procurement where the difference in bids versus the normal GSA list price was on the order of 35 percent. The evaluated difference in price was about 9 million dollars. These kinds of savings are available through competition. Such savings argue for not simply buying from some schedule, but rather taking a longer time to buy with competition being a money saver.

Workload Realignment. Based upon the graphing of the consumption of our computer center resources, we should be able to shift work from peak to off peak usage.

Understanding of Maximum Potential. A mystery of data processing seems to be the continual surprise that experienced personnel have at being nearly out of capacity to perform work or at being near some agreed set of performance parameters or service level agreements. We wanted to gain in our understanding of how much room to grow we have with our current and planned equipment.

Functional Change Analysis. An unspoken desire was the ability to assess the impact of certain new functional changes to the installed systems which support the major users of our data center, and to assess how performance of these new functions changed as the workload experienced rapid increases.

Budget and Project Planning. It is often the case that people who are in the business of creating capacity planning reports think that it is someone else who must use the reports or draw conclusions from them. We usually refer to these unseen or unidentified people as "top management" or use some other pointer that says "it isn't me that is holding us back from using this great data." Well, in fact, it is often the case in federal agencies that budget information and planning information flow upward from one with the information to one who will commit the funds and direct a project to proceed. In the U.S Customs Service, as in most business enterprises, the individual with well-thought-out information is in a position to make a positive contribution to senior level decisions. Often this is not through the actions of others up some mysterious chain of command but rather through our own actions. That is, we are often that unseen higher authority that needs to act on the information that we possess or can obtain by writing well-founded recommendations.

In summary, the process of building a capability to assess present and future usage of data processing resources has been difficult to implement. There are as many managerial problems as there are technical ones. The problems of clear communications where the technician can understand the intent of the Customs manager were a beginning obstacle. The reverse is of course also true of the manager coming to grips with his own changing desires for the project. This is the start of an agency's effort. The ultimate utility of the project will only be assessable after some years in operation.

6. FUTURE WORK

Work is proceeding on two fronts. Business identifiers are being used to present the kind of information we have discussed above for separate business categories of work. The effect of changes in the business workload can then be better assessed for its impact on the system [KRA-86]. We are also working on useful definitions of system capacities. The system reported on in this paper has not operated under a saturating workload. We hope to track the system as the workload increases and to support management in their decisions relative to the system.

7. REFERENCES

[BRO-77] Bronner,L. "Capacity Planning an Introduction," IBM Washington Systems Center, Gaithersburg, MD, IBM Corporation, January 1977, pp 24-31.

[BRO-79] Bronner,L. "Capacity Planning Implementation," IBM Washington Systems Center, Gaithersburg, MD, IBM Corporation, January 1979, pp 31-113.

[CHA-83] Chamber,J.M. et al, "Graphical Methods for Data Analysis," Duxbury Press, 1983.

[COR-83] Cortada,J.W. "Managing DP Hardware. Capacity Planning, Cost Justification, Availability, and Energy Management," Prentice Hall, 1983 pp 51-101.

[FRI-86] Friedman,E. "Modeling CICS Systems: The Performance Impact of MRO Implementations and Data Base Consideration," Proceedings of the CMG86, Las Vegas, Dec 1986, pp 82-88.

[HOA-83] Hoaglin,D. editor, "Understanding Robust and Exploratory Data Analysis," Wiley, 1983

[KRA-86] Kraslawsky,W.P. "Capacity Management Evolution at the Library of Congress," Proceedings of the CMG'86, Las Vegas, December 1986, pp 409-415.

[MER-84] Merrill,H.W., "Merrill's Expanded Guide to Computer Performance Evaluation Using the SAS System," SAS Institute, 1984.

[MOR-85] Morino Associates, Inc., "MVS Integrated Control System (MICS)," Morino Associates, Vienna Virginia, 1985.

[MOS-77] Mosteller,F and Tukey,J.W. "Data Analysis and Regression," Addison-Wesley, 1977.

[PAT-85] Paterson,G. "Computing end user Response Time in a Production CICS Network," Proceedings of the CMG85, Dallas, Dec 1985, pp 259-267.

[RMF-87] IBM Corporation, "MVS/Extended Architecture, Resource Management Facility and User's Guide," LC28-1138.

[ROS-84] Applied Data Research Inc. "ADR/ROSCOE Terminal User Guide," SR20-20-11, July 1984.

[RPF-84] Applied Data Research Inc. "ADR/ROSCOE RPF Language Manual," SR20-40-10, July 1984.

[SAS-85] SAS Institute Inc, "SAS User's Guide: Basics,", Version 5 Edition, 1985.

[SMF-87] IBM Corporation, "MVS/Extended Architecture, System Programming Library Management Facilities," GC38-1153.

[TUK-77] Tukey,J.W., "Exploratory Data Analysis," Addison-Wesley, 1977, chapter 2,3,4,7.

[WON-86] Wong,F.A. and Pedrina,F. "Automated Forecasting of Large and Complex Workloads," Proceedings of the CMG86, Las Vegas, Dec 1986, pp 121-132.

US CUSTOMS SERVICE
CPU BUSY STATISTICS
COMPUTER='3090-200'

DATE=TUESDAY, MARCH 24, 1987 CPU=CPU_1

BAR CHART OF CPU BUSY PERCENT

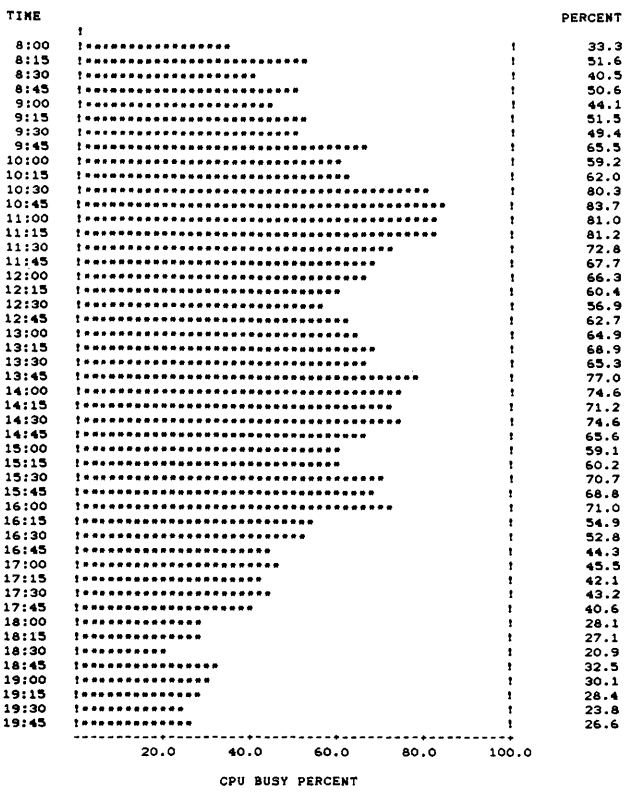


Figure 1.

US CUSTOMS SERVICE
COMPUTER RESOURCES DELIVERED
COMPUTER='3090'

COEFFICIENTS FOR SYSTEM RESOURCE UNIT CALCULATION
CPU - 10 SRB - 10 I/O - 10 MEMORY - 3
SRUS = CPUCOEFF * CPU TCB SECONDS * HARDWARE CONSTANT
+ SRBCOEFF * CPU SRB SECONDS * HARDWARE CONSTANT
+ I/O COEFF * UNITS
+ MEMCOEFF * MEMORY (PAGE SECONDS) UNITS

DATE=TUESDAY, MARCH 24, 1987

BAR CHART OF SYSTEM RESOURCE UNITS

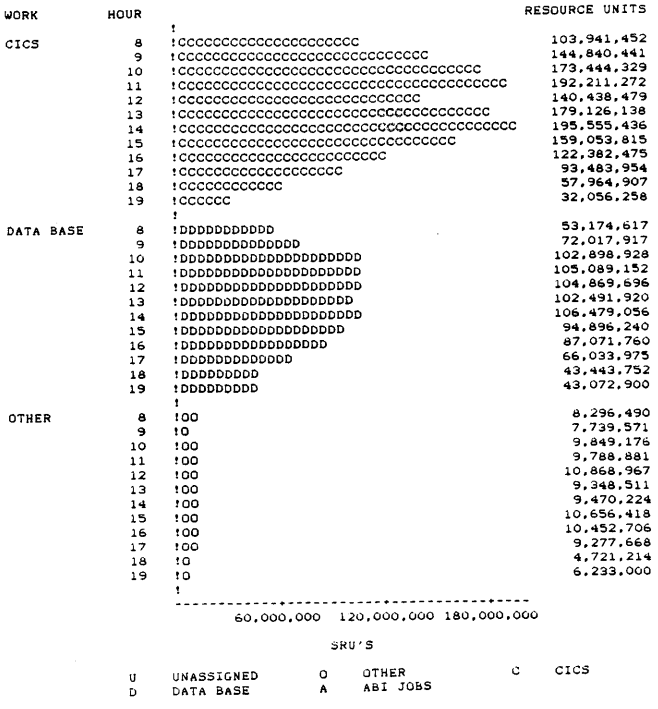


Figure 2.

US CUSTOMS SERVICE
COMPUTER RESOURCES DELIVERED
COMPUTER='3090'

COEFFICIENTS FOR SYSTEM RESOURCE UNIT CALCULATION
CPU - 10 SRB - 10 I/O - 10 MEMORY - 3

TOTUNITS = CPUCOEFF * CPU TCB SECONDS * HARDWARE CONSTANT
+ SRBCOEFF * CPU SRB SECONDS * HARDWARE CONSTANT

DATE=TUESDAY, MARCH 24, 1987

BAR CHART OF TOTAL CPU UNITS

WORK	HOUR		CPU UNITS
		!	
CICS	8	!CCCCCCCCCCCCCCCCCC	9,249,122
	9	!CCCCCCCCCCCCCCCCCCCCCCC	11,675,074
	10	!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	15,218,365
	11	!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	16,128,268
	12	!CCCCCCCCCCCCCCCCCCCCCCC	11,455,087
	13	!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	15,566,435
	14	!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	16,318,178
	15	!CCCCCCCCCCCCCCCCCCCCCCCCCC	13,728,905
	16	!CCCCCCCCCCCCCCCCCCCC	10,707,285
	17	!CCCCCCCCCCCCCC	7,888,620
	18	!CCCCCCCCCC	5,104,939
	19	!CCCCC	2,701,804
		!	
DATA BASE	8	!DDDDDDDDDDDD	6,112,536
	9	!DDDDDDDDDDDDDDD	8,078,075
	10	!DDDDDDDDDDDDDDDDDDDDDDD	11,554,361
	11	!DDDDDDDDDDDDDDDDDDDDDDD	11,633,116
	12	!DDDDDDDDDDDDDDDDDDDDDDD	11,623,865
	13	!DDDDDDDDDDDDDDDDDDDDDDD	11,418,950
	14	!DDDDDDDDDDDDDDDDDDDDDDD	11,725,664
	15	!DDDDDDDDDDDDDDDDDDDD	10,496,005
	16	!DDDDDDDDDDDDDDDDDD	9,651,208
	17	!DDDDDDDDDDDDDD	7,371,732
	18	!DDDDDDDDDD	4,886,460
	19	!DDDDDDDDDD	4,892,854
		!	
OTHER	8	!OOOOO	2,939,324
	9	!OOOOO	2,622,827
	10	!OOOOOOA	3,412,724
	11	!OOOOOOA	3,488,076
	12	!OOOOOOOA	3,979,686
	13	!OOOOOA	3,480,639
	14	!OOOOOA	3,321,654
	15	!OOOOOOA	3,628,447
	16	!OOOOOOA	3,700,491
	17	!OOOOOA	3,226,833
	18	!OOOA	1,772,069
	19	!OOOOA	2,245,956
		!	
OVERHEAD	8	!SSSSSSSSSS	4,849,669
	9	!SSSSSSSSSS	5,149,013
	10	!SSSSSSSSSSSSSSS	7,570,660
	11	!SSSSSSSSSSSSSSSSSS	8,981,901
	12	!SSSSSSSSSSS	5,852,826
	13	!SSSSSSSSSSSS	6,488,237
	14	!SSSSSSSSSSSS	6,656,347
	15	!SSSSSSSSSSSS	6,580,825
	16	!SSSSSSSSSS	5,884,266
	17	!SSSSSSSSS	4,916,073
	18	!SSSSSS	3,164,025
	19	!SSSSSSSSSS	5,435,279
		!	

6,000,000 12,000,000 18,000,000

TOTAL CPU UNITS

KEY TO SYMBOLS FOR THE BARS

U	UNASSIGNED	O	OTHER	C	CICS
D	DATA BASE	A	ABI JOBS	S	SYSTEM OVERHEAD

Figure 3.

US CUSTOMS SERVICE
CICS RESPONSE TIME STATISTICS
STATISTICS FOR 8AM TO 8PM

PRODUCTION WORKLOAD COMPUTER='3090-200'

TUESDAY, MARCH 24, 1987

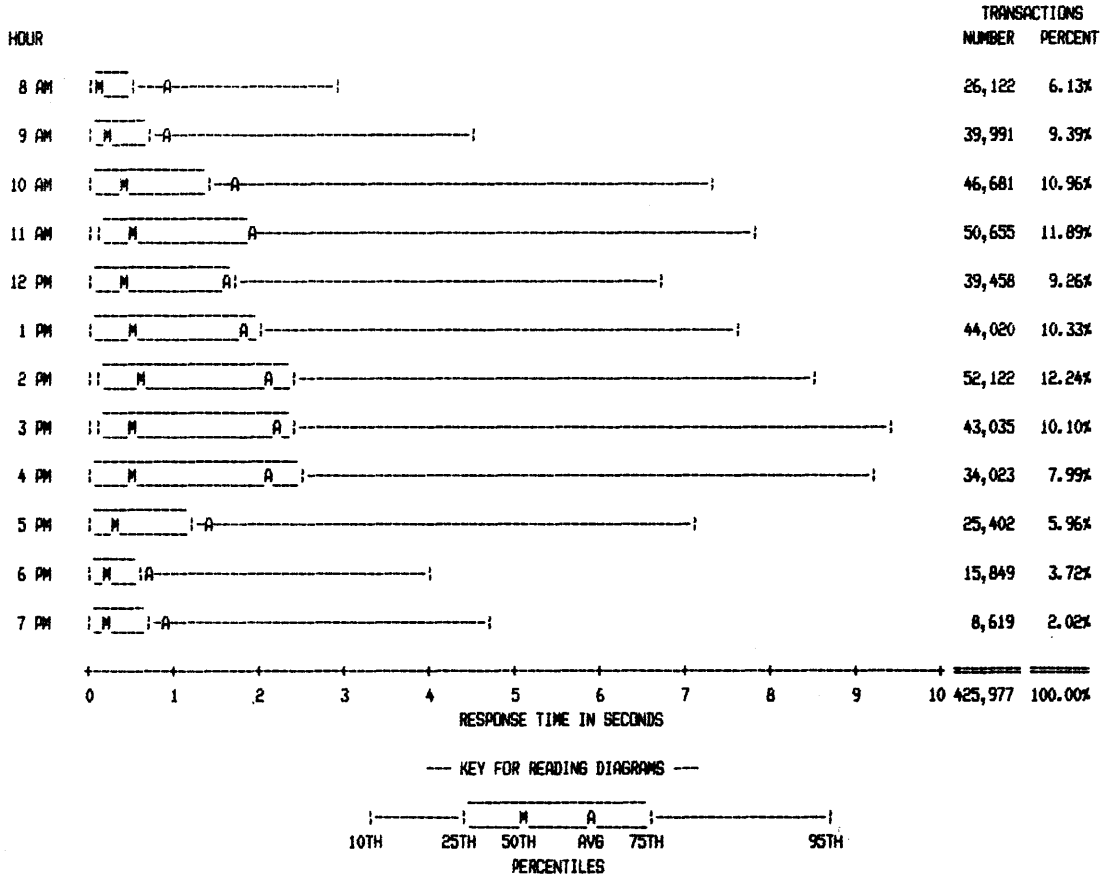


Figure 4.

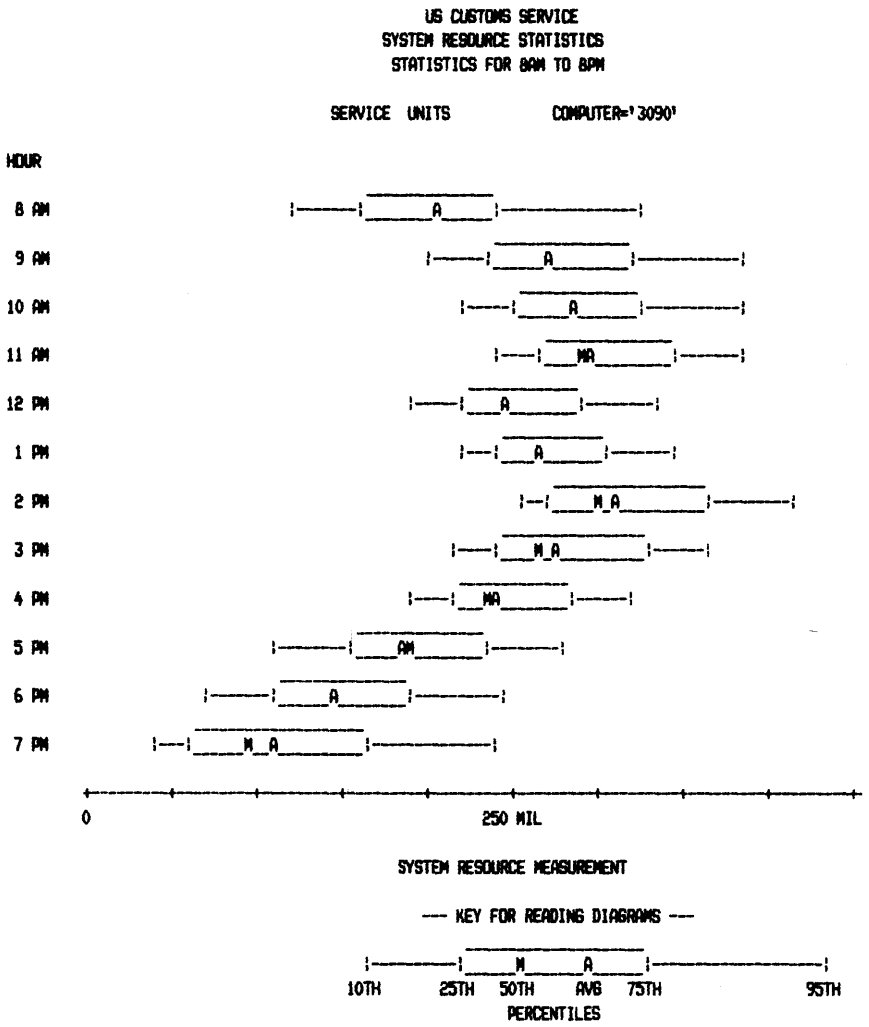


Figure 5.

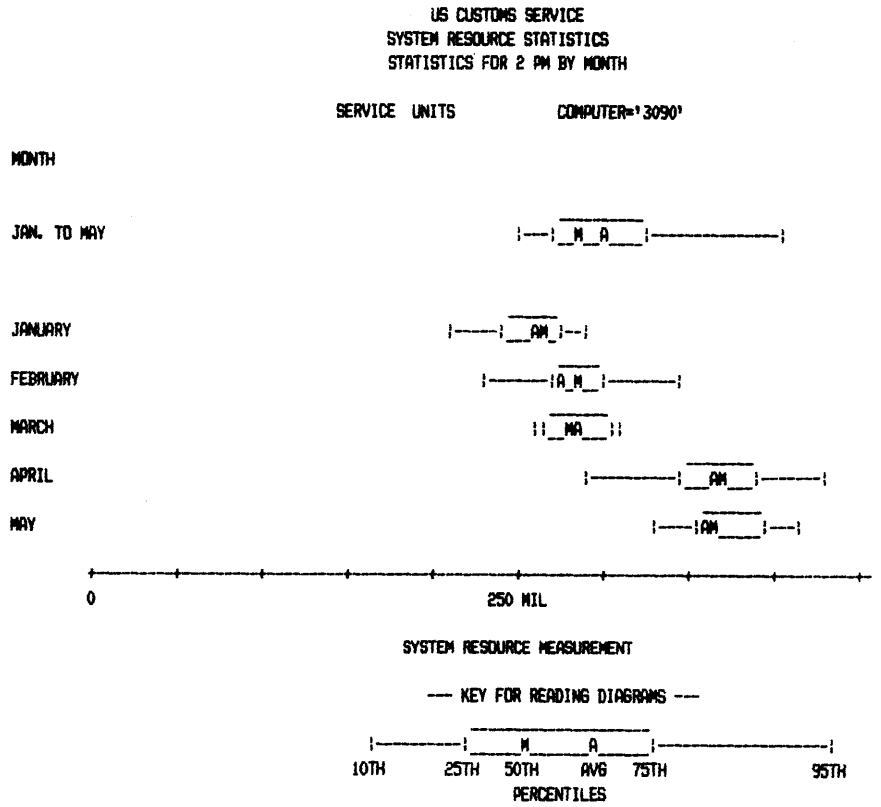


Figure 6.

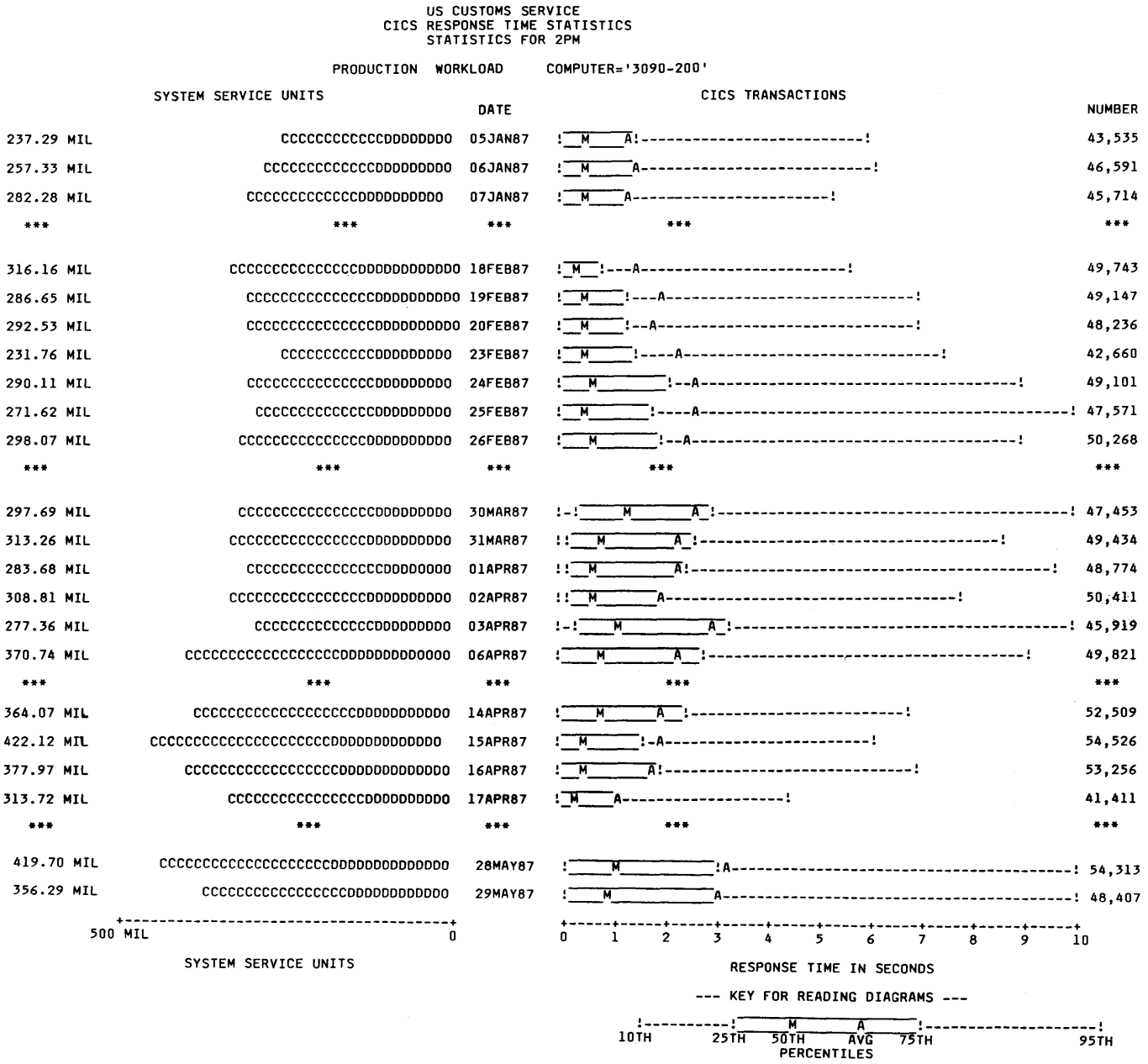

US CUSTOMS SERVICE
CICS RESPONSE TIME STATISTICS
STATISTICS FOR 2PM
PRODUCTION WORKLOAD COMPUTER='3090-200'
SYSTEM SERVICE UNITS
DATE
CICS TRANSACTIONS
NUMBER
237.29 MIL 05JAN87 43,535
257.33 MIL 06JAN87 46,591
282.28 MIL 07JAN87 45,714
316.16 MIL 18FEB87 49,743
286.65 MIL 19FEB87 49,147
292.53 MIL 20FEB87 48,236
231.76 MIL 23FEB87 42,660
290.11 MIL 24FEB87 49,101
271.62 MIL 25FEB87 47,571
298.07 MIL 26FEB87 50,268
297.69 MIL 30MAR87 47,453
313.26 MIL 31MAR87 49,434
283.68 MIL 01APR87 48,774
308.81 MIL 02APR87 50,411
277.36 MIL 03APR87 45,919
370.74 MIL 06APR87 49,821
364.07 MIL 14APR87 52,509
422.12 MIL 15APR87 54,526
377.97 MIL 16APR87 53,256
313.72 MIL 17APR87 41,411
419.70 MIL 28MAY87 54,313
356.29 MIL 29MAY87 48,407
500 MIL
0
SYSTEM SERVICE UNITS
0 1 2 3 4 5 6 7 8 9 10
RESPONSE TIME IN SECONDS
--- KEY FOR READING DIAGRAMS ---
10TH 25TH 50TH AVG 75TH 95TH
PERCENTILES

Figure 7.

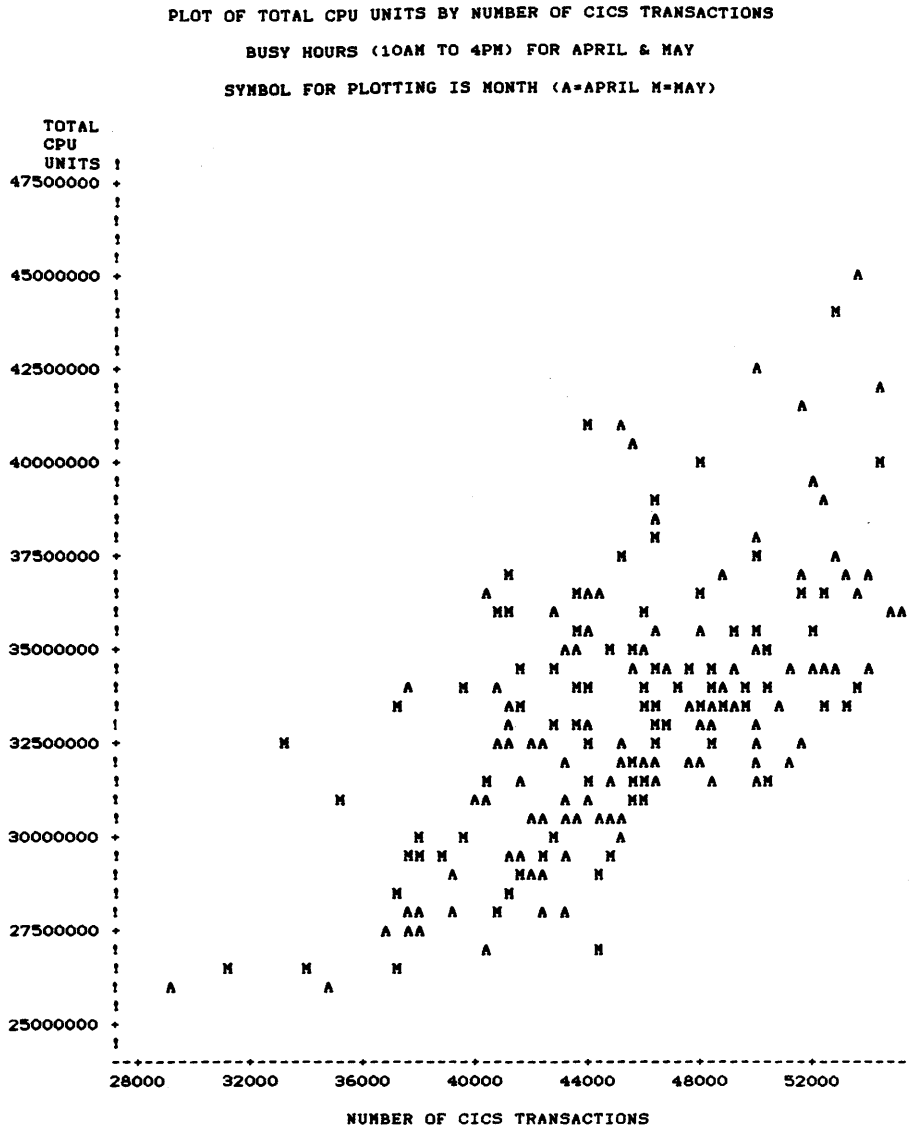


Figure 8.

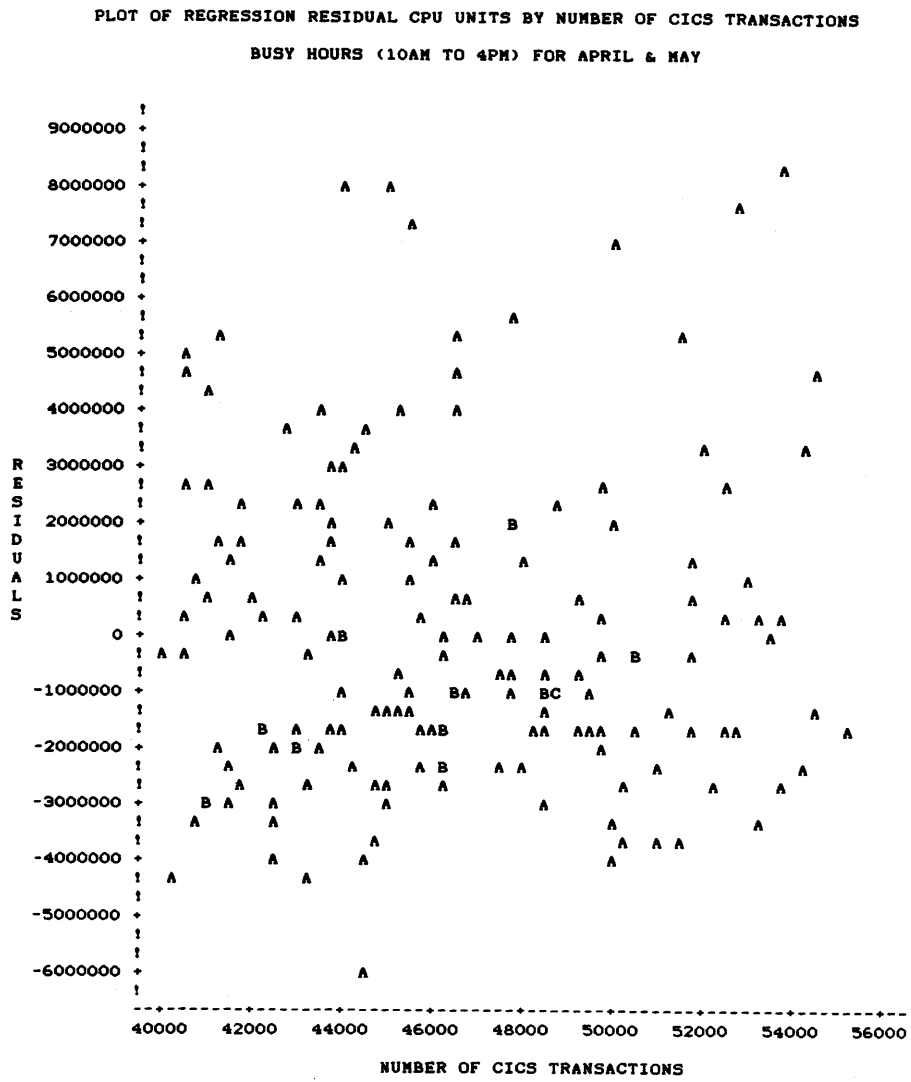


Figure 9.

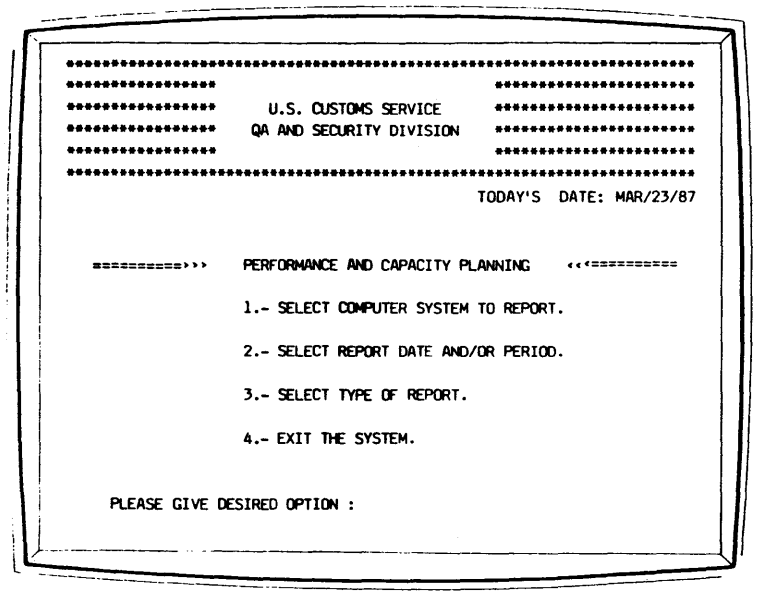

U.S. CUSTOMS SERVICE
QA AND SECURITY DIVISION
TODAY'S DATE: MAR/23/87
PERFORMANCE AND CAPACITY PLANNING
1.- SELECT COMPUTER SYSTEM TO REPORT.
2.- SELECT REPORT DATE AND/OR PERIOD.
3.- SELECT TYPE OF REPORT.
4.- EXIT THE SYSTEM.
PLEASE GIVE DESIRED OPTION :

Figure 10.

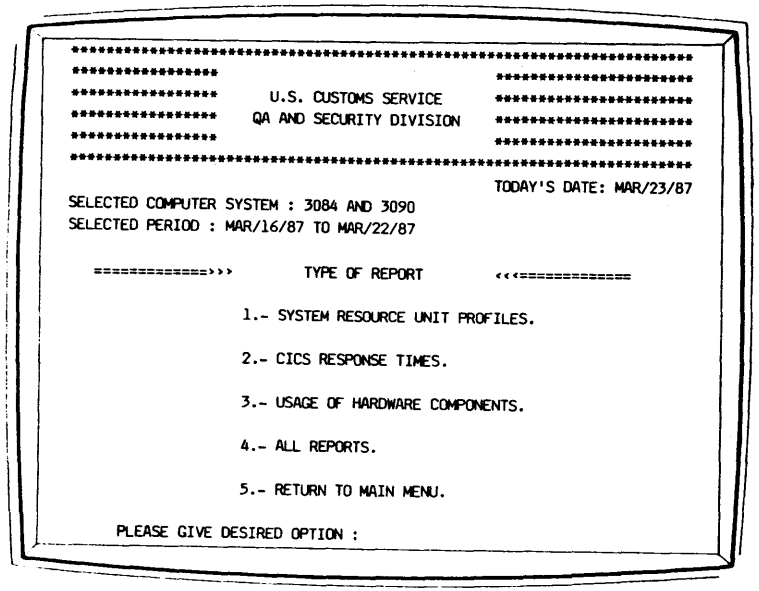

U.S. CUSTOMS SERVICE
QA AND SECURITY DIVISION
TODAY'S DATE: MAR/23/87
SELECTED COMPUTER SYSTEM : 3084 AND 3090
SELECTED PERIOD : MAR/16/87 TO MAR/22/87
TYPE OF REPORT
1.- SYSTEM RESOURCE UNIT PROFILES.
2.- CICS RESPONSE TIMES.
3.- USAGE OF HARDWARE COMPONENTS.
4.- ALL REPORTS.
5.- RETURN TO MAIN MENU.
PLEASE GIVE DESIRED OPTION :

Figure 11.

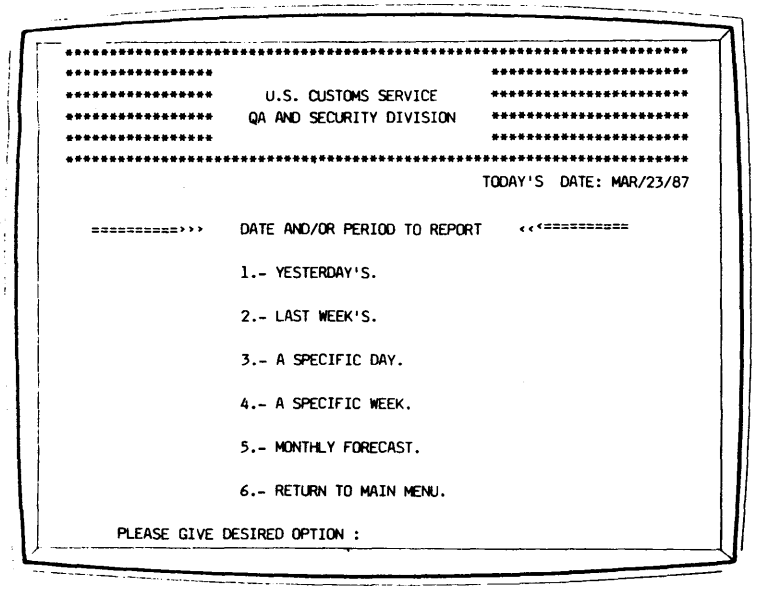


Figure 12.

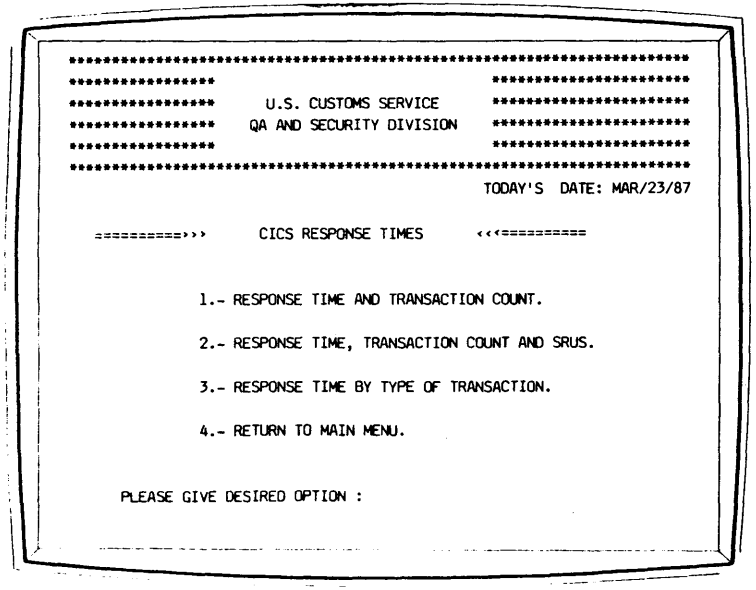

U.S. CUSTOMS SERVICE
QA AND SECURITY DIVISION
TODAY'S DATE: MAR/23/87
CICS RESPONSE TIMES
1.- RESPONSE TIME AND TRANSACTION COUNT.
2.- RESPONSE TIME, TRANSACTION COUNT AND SRUS.
3.- RESPONSE TIME BY TYPE OF TRANSACTION.
4.- RETURN TO MAIN MENU.
PLEASE GIVE DESIRED OPTION :

Figure 13.

MODEL TIMEFRAME SELECTION AND ANALYSIS OF VARIANCE OF AN ONLINE TRANSACTION PROCESSING WORKLOAD

By
Theresa A. Hellner and Dennis J. Henry
American Express Co.
Travel Related Services

ABSTRACT

This paper will demonstrate a methodology for selecting a timeframe to represent the baseline model of a central processor complex dominated by an online transaction processing workload such as IMS, or CICS. The final step in modeling a given workload is the verification process, which compares the predicted to actual results. In order to manage a forecast over it's expected lifespan, a methodology to account for the variances between the predicted and actual results is necessary. A practical application of David Vincent's methodology entitled, *"Analysis of Variance of Workloads"*, which appeared in the October 1986 issue of the EDP PERFORMANCE REVIEW is demonstrated. In closing, the authors draw a correlation between selecting a timeframe to model and the verification process. Worksheets and sample code to perform the necessary calculations for the analysis of variance are also included.

INTRODUCTION

This paper demonstrates a practical implementation of David Vincent's methodology entitled *"Analysis of Variance of Workloads"* published in the October 1986 issue of the EDP Performance Review. This method is used to account for variances in a processor complex dominated by online transactions, namely IMS. For several years the Capacity Planning department has tracked the overall actual to forecast variance of total processor utilization and IMS transaction counts. When it became necessary that a more detailed tracking of the IMS application workload was required, variance analysis was undertaken and the use of Information Planning Units (IPU) was implemented. Besides variance in transaction counts and application CPU usage, the transaction mix (arrivals and CPU time by workload) can now be accounted for. This implementation is demonstrated by the following topics:

- Representative timeframe selection
- Overview of the modeling process
- Analysis of variance using IPUs
- Practical application using an actual case
- Correlation of timeframe selection and verification
- DB2 IPU development

Capacity Planning is responsible for various data centers with IBM 3090-400 and 3090-200 processors and several smaller processors. Due to the complexity of the these centers, Capacity Planning's charter is to forecast only processor growth. I/O Configurations are the responsibility of the data center technical support staffs.

The major workload for this discussion is IMS which is the driver of specific processor upgrades and accounts for approximately 50% to 85% of the prime time processor utilization for the processors that host the IMS Subsystems. The IMS Subsystems are split by application, hence, forecasting can be done using one or two primary IMS applications as drivers of a subsystem. Data centers that forecast using IMS transaction classes can also apply the following procedure.

REPRESENTATIVE TIMEFRAME SELECTION

The following discussion concerns selecting a model timeframe for a processor whose upgrades are driven by a large IMS Subsystem. However, the same methodology can be applied to a processor dominated by other workloads. The various jobs submitted by the analyst are, in this example, based on data files in a MICS Database and indicate both the transaction volume and transaction workload mix by IMS Subsystem ID. When more than one IMS Subsystem is on a processor, the IMSID's are combined. All of the programs discussed in this paper were developed in-house by the Capacity Planning staff.

The direction given to Capacity Planning for sizing the processors is based on a representative peak processing hour, not the peak hour or the 90th percentile. The representative peak hour is actually an average of the peak processing hour of each day over a one month period of time. Days with outages during the peak volume period of Time Zone 1 are eliminated from this calculation.

Selection of a representative day and hour begins with a calendar report which shows total IMS activity within the prime shift (Time Zone 1). This program is the only one where multiple IMSID's executing on the same processor are combined to show the busiest hour by transaction volume. Days that incurred outages are noted on this report and excluded from any further processing. This report (see Figure 1) indicates by day the following information:

- total transaction volume for Zone 1
- peak hour transaction volume
- peak hour of occurrence
- CPU msec/transaction across Zone 1
- average transactions/second

APRIL 1987

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	
27	28	29	30		LEGEND
552523	553688	535275	507696	---	Total trans
95955	95995	98450	93479	---	Peak Trans/hr
18.1	18.1	17.9	18.2	---	CPU ms/tran
25.6	25.6	24.8	23.5	---	Trans/sec
11	11	10	11	---	Peak hour

Figure 1. Partial listing of the Calendar Report

90th % AND AVERAGE FOR TRANSACTON VOLUME,
CPU TIME/TRAN AND RESPONSE TIME/TRAN

FOR ALL ZONE 1 HOURS BY WORKLOAD

HOUR	WKLD	TRANS1	CPUTM1	TRSTM1	AVGTRANS	PCTRANS	AVGCPUTM	PCTCPUTM	AVGTRSTM
10	AR	78053.6	0.0215	0.75	73245.9	82.85	0.0085	91.96	0.46
10	OTR	17021.0	0.0146	0.64	15161.9	17.15	0.0076	08.04	0.37
11	AR	81107.0	0.0217	0.75	76474.2	82.63	0.0207	92.01	0.53
11	OTR	18210.5	0.0145	0.62	16075.9	17.37	0.0084	07.99	0.32

Figure 2. Example of the monthly report by hour

The function of the second program is twofold. First, it produces an hourly profile of the major applications by IMSID within Time Zone 1 for the entire month. The premise is that the IMS Subsystem has one or two major applications that represent the majority of the total workload and therefore actually drives the IMS forecast. The second function is to produce a list of the potential timeframes to model. To find a representative day in terms of workload mix, the analyst must come close to matching the applications's transaction volume, and CPU consumption. Looking at one or two applications per IMS Subsystem saves time in selecting the correct day and timeframe to model.

The MICS daily IMSIAU file is used as the input. Figure 2 shows each IMSID and the major applications 90th percentile hourly transaction volume (TRANS1), and average hourly transaction volume (AVGTRANS). Figure 2 also shows the percentage of total transactions (PCTRANS), and the percent CPU busy time (PCTCPUTM) that the major applications required in relation to that of the entire IMS Subsystem. Other information on this report is the 90th percentile and average CPU time (CPUTM1 and AVGCPUTM) and response times (TRSTM1 and AVGTRSTM) per transaction for each of the major applications. As can be seen this report shows clearly that the AR application is the dominant application of this IMS Subsystem accounting for nearly 83% of the total transactions and 92% of the total CPU time over each hour of Time Zone 1.

The second report (Figure 3) produced by this program is a list of selected representative peak hours that are potential candidates for the baseline model. For each IMS Subsystem the program creates a transaction volume range based on the average peak hour transaction volume over the month. The high and low values of the range are then calculated as plus or minus 1% of the average peak volume. Hourly transaction volumes within this range are then listed in the report.

Choosing the representative hour requires the analyst to also validate that the selected timeframe does not include any system outages and that the application transaction mix is typical of the IMS Subsystem daily performance. This is verified by executing a detailed report for the day and hour chosen from Figure 3 and comparing its output to the hourly transaction volumes, CPU time per transaction, and response time per transaction for the major applications within the IMSID to that in Figure 2. The percentage of total transactions and CPU time of the applications will also be compared to Figure 2.

The particular report that is used to validate a potential timeframe to model is shown in Figure 4. The timeframe selected from Figure 3 was April 28, 1987 for the hour of 10:00. The AR application profile for this particular hour is the only one that the analyst will try to match up with the hour of 10:00 on Figure 2. As you can see the AR CPU time per transaction of 20.22 milliseconds on Figure 4 matches up very well with the AVGCPUTM of 20.3 milliseconds for the selected hour. The average response time of .46 seconds in Figure 4 matches to the AVGTRSTM of .46 seconds on Figure 2. The percentage of transactions and the percentage of CPU time that the AR application represent, 84.9% and 93.5% respectively, of the total IMS Subsystem on Figure 4 are slightly higher than the average of 82.85% and 91.96% shown on Figure 2. Since this day is a representative peak timeframe that difference can be attributed to transaction volume.

The key here is that the driving application's (AR) profile for this particular IMS Subsystem during this timeframe fits very nicely with the profile for the hour of 10:00 shown in Figure 2. This timeframe is a good candidate for the baseline model.

SELECTED REPRESENTATIVE PEAK HOURS

IMSID	DATE	HOUR	IMSLOVAL	IMSHIVAL	IAUTRANS	IAUTRSTM	IAUCPUTM
IMS	28APR87	10	91821.1	93676.1	92693	0.4418	0.0183
IMS	28APR87	15	91821.1	93676.1	93431	0.3999	0.0182
IMS	30APR87	11	91821.1	93676.1	93479	0.3816	0.0187

Figure 3. Example of the representative peak hour report

APPLICATION USAGE SUMMARY

IMS TUESDAY APRIL 28,1987 HOUR = 10

APPL	TOTAL TRANS	CPU (HMS)	MSEC/ TRANS	AVG RESP
AR	78,652	0:26:30	20.22	0.46
OTR	14,041	0:01:50	7.80	0.33
TOTAL	92,693	0:28:20	18.34	0.44

PERCENT TRANS	PERCENT CPU
84.9	93.5
15.1	06.5

Figure 4. Example of IMS application report

OVERVIEW OF THE MODELING PROCESS

Once the representative hour is selected for a processor, the baseline model is created using BEST/1 or another modeling system. The processor utilization, arrival rates, and the IMS average response times are validated. The forecast is then applied to project growth over the required timeframe (for example two years).

Capacity Planning gathers forecast data using expected normal business growth, the addition of new IMS applications, and anticipated growths of existing terminal users. Forecasts are not based on historical data but do include seasonality concerns. The model is then iterated over the forecast duration on a periodic basis which is monthly. The forecast is maintained in a BEST/1 command list which simplifies adjustments and multiple iterations.

Two forms of the model are executed. The first is an unconstrained version with all possible I/O required, and any additional processor upgrades necessary to maintain response time. All workloads are kept during the iterations. This provides information with regards to potential constraints due to the execution of the existing total workload (IMS and batch). Preparation for this version may require several iterations of the model prior to the full blown unconstrained version to identify a potential iteration (month) which may need adjustment to maintain the unconstrained goal.

A second version of the model is executed using the current processor model to determine workload break points. I/O upgrades that were needed in the unconstrained version are maintained. This version is used to determine if the IMS response time exceeds the acceptable limits, what effects removing workloads are to the forecast, and how and when other workloads on the processor degrade performance.

Upgrades are projected for the quarter in which the model iteration exceeds requirements. Based on the results of the second version of the model, recommendations are then presented to management concerning processor upgrades, workload shifts between processors, and warnings concerning service level impacts.

The results of the constrained model, the second version, are kept in the generated BEST/1 SAS file and forms the basis for the forecasted/planned IPUs that will be used in the analysis of variance procedure.

IMS 4/28/87 HOUR = 10

ISYTRANS	ISYCPUTM	ISYCPTRN	TRSTM	WKLPGATM	CRATIO	ADJPGATM
92693	0:28:20.4	.0183	0.442	0:47:31.2	0.865	0:54:54.52

CPU_TRAN	ADJISY	PCTBUSY	PCTWKLD
0.355	51.61 %	98.97 %	45.75 %

Figure 5. Workload for Representative Hour

ANALYSIS OF VARIANCE USING INFORMATION PLANNING UNITS

An Information Planning Unit (IPU), as defined by Vincent, is the total system resource required to process each transaction. In the following implementation, this includes only the transaction volume and the CPU time adjusted by the capture ratio for the representative hour. The forecasted/planned IPU is calculated from the model iteration. The *Plan Unit* is the forecasted transaction volume (arrival rate) for a particular iteration of the model. The *Plan IPU* is the *Plan Unit* times the average CPU time for a transaction taken from the baseline model or the model iteration if a forecast adjustment was made during the iteration. An average *Plan IPU* is then simply the expected average CPU time per transaction.

The *Actual IPU* is taken from the observed IMS application performance for the day and hour chosen by the representative hour method described previously. The *Actual Unit* is the current transaction volume and the *Actual IPU* is the *Actual Unit* times the current average CPU per transaction for the application or overall IMS Subsystem. Since the forecast is based on monthly intervals the analysis of Actual to Planned variances is on a monthly basis.

Three levels of analysis are undertaken: Volume Variance, Transaction Mix Variance, and a Rate Variance. The Volume Variance is that which occurs when the transaction volume for the IMS Subsystem is greater or less than that which was forecasted. The Mix Variance results when the application mix within the IMS Subsystem differs from that forecasted and again uses the total IMS data values. In this case the overall average CPU time is used in calculating the IPU values for Planned and Actual. Both of these variances are related to the user requests for service and are stated in Figure 6.

The third variance is reflective of the resource usage required by the IMS application. Unlike the previous variances the Rate Variance is influenced by system capacity, performance, or inefficient transaction design. The Rate Variance for this implementation occurs when the *Actual IPU* varies from the *Planned IPU* on an application basis within the IMS Subsystem. This then is the summation of the primary two application IPU differences multiplied by the observed *Actual Units* for the applications and the IPU for all the other IMS workloads multiplied by the total *Actual Units* for the other workloads within the IMSID. This is expressed as the Rate Variance in Figure 6.

The TOTAL Variance for the IMS system is the sum of the Volume, Mix, and Rate variances. For reporting purposes the variances are expressed as the percent difference of the Actual and Planned IPU performance.

PRACTICAL APPLICATION USING AN ACTUAL CASE

The following demonstration of variance analysis of an IMS Subsystem is based on comparing a forecast of the Subsystem done in November of 1986 to actual resources consumed in April 1987. Because the forecast iterations are based on monthly intervals, the verification of the forecast begins with the selection of the representative peak processing hour for the month of April. In a previous section of this paper it was proved that April 28, 1987 from 10:00 to 11:00 was a representative peak hour for the month. The remainder of this section will calculate the variance of the forecasted and actual resources consumed and how much of that can be attributed to Transaction Volume, Transaction Mix, and the CPU rate per transaction.

In order to determine the workload variance the necessary information is first written down on a worksheet (Attachment A) and is then entered into a SAS program as Macros (Attachment B). This program calculates the total variance and apportions the variance of the three variables, transaction volume, transaction mix, and transaction CPU usage or rate per transaction Figure 7.

The first item to be filled out on the worksheet is **_PLANIPU.** This is the Planned Information Planning Units. In this case the CPU utilization of the IMS Subsystem from the BEST/1 forecast was converted into seconds. For April of 1987 our forecast showed the IMS Subsystem would consume 3,188.9 CPU seconds during a representative peak hour.

_PLANUNI is the forecasted transaction arrival rate also obtained from the BEST/1 model for April 1987. The forecast showed that the transaction volume of a representative peak hour would be 92,880.

_ACTLUNI is the actual transaction volume from the selected representative peak hour during April 1987 obtained from the Application Usage Summary report (Figure 4). The actual transaction volume was 92,693.

_ACT1UNI is the actual transaction volume of the primary workload of the IMS Subsystem for April 1987 obtained from the Application Usage Summary report (Figure 4). Since one workload is dominant in terms of transaction volume and CPU usage, _ACT2UNI was set at zero. The actual transaction volume for the primary workload was 78,652.

$$\text{Volume Variance} = \frac{\text{Plan IPU}}{\text{Plan Unit}} \times (\text{Actual Unit} - \text{Plan Unit})$$

$$\text{Mix Variance} = \left(\frac{\text{Actual IPU}}{\text{Actual Unit}} - \frac{\text{Plan IPU}}{\text{Plan Unit}}\right) \times (\text{Actual Unit})$$

$$\text{Rate Variance} = \text{SUM}((\text{ACTIPU1} - \text{PLNIPU1}) \times (\text{Actual Units 1}), (\text{ACTIPU2} - \text{PLNIPU2}) \times (\text{Actual Units 2}), (\text{ACTIPUn} - \text{PLNIPUn}) \times (\text{Actual Units n}))$$

Figure 6. Variance Equations for IPUs

_ACT1IPU is the actual primary workload CPU time per transaction from the selected representative peak hour expressed in milliseconds. This is calculated by dividing the primary workload milliseconds per transaction from the Application Usage Summary report (Figure 4) by the ratio of ISYCPUTM to the adjusted workload (ADJISY) performance group time (Figure 5). The ratio for the selected representative peak hour was 51.61%. This means that the ISYCPUTM from the MICS IMS database for this selected representative peak hour was 51.61% of the IMS performance groups PGACPUTM adjusted with a capture ratio (ADJPGATM, CRATIO). The actual primary workload CPU time per transaction for April 1987 was .03918. _ACT2IPU was set at zero.

_ACTXIPU is the actual total non-primary workload CPU time in milliseconds from the selected representative peak hour. The method for calculating this value is the same as for _ACT1IPU with the exception of subtracting out the primary workload(s) CPU time. This number is then divided by the number of non-primary workload transactions. In our example the system total milliseconds per transaction from the Application Usage Summary report was divided by 51.61% (ADJISY). This value was multiplied by the total number of transactions to give a total CPU time of the IMS Subsystem. Next, the _ACT1IPU was multiplied by _ACT1UNI. This CPU time which represents the primary workload was then subtracted from the total CPU time for the IMS Subsystem. The remaining non-primary workload CPU time was then divided by the non-primary transactions, _ACTLUNI minus _ACT1UNI. The actual non-primary workload CPU time per transaction for April 1987 was .01515.

_PLN1IPU is the planned primary workload CPU time in milliseconds per transaction. This number is derived by multiplying _PLANIPU by the percentage of total IMS Subsystem CPU time the primary workload is expected to be at this point of the forecast. This information is extrapolated from the selected hour used in the baseline model plus or minus any adjustments made over the forecast. Since there is only one primary workload, _PLN2IPU is zero. The planned primary workload CPU time per transaction for April 1987 is .0379 milliseconds.

_PLNXIPU is the planned non-primary workload CPU time per transaction in milliseconds. It is calculated the same way as _PLN1IPU but for non-primary workload transactions only. The planned non-primary workload CPU time per transaction for April 1987 is .0137.

The output from the program used to determine the workload variance of the IMS Subsystem for April 1987 is shown on Attachment A. The total variance of Actual IPUs to Planned IPU was 3.3%. Of this total variance, -0.2% was attributable to a slightly lower actual transaction volume than was planned. Transaction Mix accounted for -0.3%, and CPU usage or rate per transaction accounted for 3.8%, due to a higher actual CPU rate per transaction versus the planned CPU rate per transaction for both the primary and non-primary workloads. All initial conclusions and recommendations made at the time of the original forecast for this particular IMS Subsystem would appear to still be valid.

The final verification of that is dependent on whether or not service levels are being met. In this case, response time is synonymous with service levels. If the forecasted response time at a particular model iteration is at or under the service level agreement and the actual response time meets accepted service levels then the original forecast remains valid.

CORRELATION OF MODEL TIMEFRAME SELECTION AND VERIFICATION

Even though the forecast is currently done on a periodic basis the entire process of model timeframe selection and forecast verification is a continual process. Capacity Planning verifies the IMS forecasts on a monthly basis. The same process that was used to select the timeframe for the baseline model is used to verify the forecast. At a future date, when it is time to generate a new forecast, the verification process provides the analyst with the current baseline.

WORKLOAD VARIANCE ANALYSIS
ACTUAL VERSUS PLANNED IMS CPU USAGE
IMS SUBSYSTEM APRIL 1987

PLANNED INFORMATION PLANNING UNITS FORECAST FOR A
REPRESENTATIVE PEAK HOUR: TOTAL CPU TIME IN SECONDS = **3188.9**

ACTUAL INFORMATION PLANNING UNITS CONSUMED FOR A
REPRESENTATIVE PEAK HOUR: TOTAL CPU TIME IN SECONDS = **3294.3**

PERCENTAGE OF VARIANCE DUE TO
TRANSACTION VOLUME ..... **-0.2**

PERCENTAGE OF VARIANCE DUE
TO TRANSACTION MIX ..... **-0.3**

PERCENTAGE OF VARIANCE DUE TO
CPU USAGE OR RATE PER
TRANSACTION **3.8**

TOTAL VARIANCE **3.3**

* A negative variance indicates the actual results were under the forecast.

Figure 7. IPU calculation results

DB2 IPU DEVELOPMENT

Development of IPUs for a DB2 Subsystem is currently in progress. DB2 is operated under TSO/QMF with the installation of an IMS interface expected in the near future. The DB2 Subsystem shares the processor with various other workloads. It is not the heaviest workload on the processor, but due to the user support functions, it is the most critical with regards to the need for sufficient processing resources. The DB2 Subsystem does not maintain any consistency in the number of queries, processor utilization, table processing counts, etc., and is continuously growing in all of the above areas.

As can be expected from this, the peak to average ratios for query processing form a wide range. These ratios are tracked using MICS for both performance group RMF statistics, (SCPPGA file) and the DB2 SMF log facilities (DB2DSY file). The later is used to define an average query per thread using the CPU time, and the table access counts (select, fetch, sequential prefetch, update, insert and delete). Since threads can be reused, they are not indicative of arrival rates and are used only as additional information for selecting the DB2 representative hour.

Selection of the model timeframe (day and hour) for this processor is based on the ninetieth percentile query estimate using the query termination plus abort count from the MICS DSY file (DSYTERM + DSYABORT). Not all DB2/TSO users are in individual RMF performance groups or reporting groups, which requires that the traditional TSO users to be modeled as part of the primary workload. The actual day and hour for the model timeframe capture is selected using the previously defined method for the IMS system.

The baseline model combines the performance groups for TSO, DB2 TSO users, and the DB2 system address spaces into one workload, batch into another, etc. Workload growths are gathered for individual DB2 applications, applied to the forecast, and modeled as a TP workload using the RMF ended transactions (PGATTX) as the arrival rate.

The average response time is used simply as a guideline in the current model iteration evaluations. The response times range from several seconds to several minutes due to table sizes up to several hundred thousands and is therefore not a modeling criteria. Again the output of the BEST/1 model is saved in a SAS file for variance comparison of forecasted arrivals.

The IPUs for the DB2 workload consist of the ended transaction counts (PGATTX) and the average CPU seconds per transaction (PGACPUTM) as reported by RMF and captured in the baseline model. The Volume Variance, Mix Variance, and Rate Variance are calculated and evaluated using the same method previously defined, with exception of using the ISYCPUTM to PGACPUTM ratio. This ratio has not yet been validated for DB2 as most of the processing is charged to the user address space. The workload time (PGACPUTM) to the total PGACPUTM is used. In this case, the primary workload is defined as the TSO system (both DB2 and non-DB2 users). The secondary workload is the TSO/DB2 only performance group. All other performance groups, including the DB2 system address space, and the IRLM are combined in the last IPU grouping. The analysis proceeds as in the above IMS example. The key is in the definition of the primary/secondary workloads.

The drawback of the ended transactions from RMF is that it does not directly relate a transaction to the SQL query. In forecast data gathering, the users issue estimates on the basis of the latter.

A further exploration is being made to also include the average table I/O count as part of the definition. The IPU would then be the transaction count times the average CPU seconds plus the transaction count times the average table access count rounded to housands. The difficulty in this is defining the average table access count per workload. Therefore an overall average per thread count is used and applied to each workload in the Rate variance.

Another approach is to use the estimated I/O count from the pages read (RIO) and pages updated (WIO) calculations using the DB2 buffer statistics (in the MICS DSD file) averaged by the number of queries (DSYTERM + DSYABORT).

CONCLUSION

This paper has demonstrated the following: 1) a methodology for selecting a representative timeframe to model an online transaction processing system such as IMS. 2) An overview of the modeling process. 3) An overview of Information Planning Units (IPUs) and analysis of variance of workloads as introduced by David Vincent. 4) A practical application of IPUs and analysis of variance of an IMS workload. 5) Correlation of the model timeframe selection and forecast verification process. 6) Application of IPUs and analysis of variance of a DB2 workload.

This whole exercise of workload variance analysis is to determine whether or not a workload is consuming resources according to expectations. If it is not, then this technique can assist the capacity planner in determining what caused the variance and how this variance is impacting the forecast and ultimately the end user.

REFERENCES

BGS Systems, Inc., BEST/1 MVS Reference Guide, Release 10.0, Doc, No: 85087-00, 1986

IBM Corporation, Annotated IMS/VS Performance Reports, GG24-1559, October 1983

IBM Corporation, Database 2 Version 1 Release 2 Performance Report, GG24-3146, February 1987

IBM Corporation, Database 2 System Monitoring and Tuning Guide, GG24-3005, July 1986

IBM Corporation, IMS/VS Version 1 System Administration Guide, SH20-9178

IBM Corporation, MVS/Extended Architecture Resource Measurement Facility and Users Guide, LC28-1138

Morino Associates Inc., MICS Data Dictionary Release 4

Vincent, David R., "Analysis of Variance of Workloads", EDP Performance Review, Volume 14, Number 10, October 1986, pgs 1-5

ANALYSIS OF VARIANCE OF WORKLOADS WORKSHEET
IMS SUBSYSTEM AT ***SITE***
MONTH ***APRIL*** YEAR ***1987***

3188.9 _PLANIPU = TOTAL CPU UTILIZATION OF THE SUBSYSTEM FROM THE BEST/1 MODEL CONVERTED TO SECONDS.

92,880 _PLANUNI = FORECASTED TRANSACTION VOLUME PER SECOND FROM THE BEST/1 MODEL CONVERTED TO TRANSACTIONS PER HOUR.

92,693 _ACTLUNI = TOTAL TRANSACTION VOLUME FROM THE SELECTED REPRESENTATIVE PEAK HOUR.

78,662 _ACT1UNI = ACTUAL PRIMARY WORKLOAD VOLUME FROM THE SELECTED REPRESENTATIVE PEAK HOUR. IF MORE THAN ONE
0.000 _ACT2UNI WORKLOAD, FILL IN _ACT2UNI TO _ACTnUNI.

0.3918 _ACT1IPU = ACTUAL PRIMARY WORKLOAD MSECS PER TRAN FROM THE SELECTED REPRESENTATIVE PEAK HOUR, DIVIDED BY
0.000 _ACT2IPU THE RATIO OF ISYCPUTM OVER ADJUSTED PGACPUTM. IF MORE THAN ONE WORKLOAD FILL IN _ACT2IPU TO ACTnIPU.

0.1515 _ACTXIPU = ACTUAL TOTAL NON-PRIMARY WORKLOAD CPU TIME IN MSECS PER TRAN FROM SELECTED PEAK HOUR. CALCULATED BY DIVIDING TOTAL ISYCPUTM BY THE RATIO OF ISYCPUTM OVER ADJUSTED IMS PGACPUTM AND AND SUBTRACTING OUT THE PRIMARY WORKLOAD(S) CPU TIME. THIS NUMBER IS THEN DIVIDED BY THE NUMBER OF NON-PRIMARY WORKLOAD TRANSACTIONS.

0.0379 _PLN1IPU = PLANNED PRIMARY WORKLOAD CPU TIME IN MSECS. MULTIPLY _PLANIPU BY PERCENTAGE OF TOTAL CPU TIME THE PRIMARY WORKLOAD IS EXPECTED TO BE, AT THIS POINT IN THE FORECAST. THIS INFORMATION
0.000 _PLN2IPU IS EXTRAPOLATED FROM THE SELECTED HOUR USED IN THE BASELINE MODEL PLUS OR MINUS ANY ADJUSTMENTS MADE OVER THE FORECAST. IF MORE THAN ONE WORKLOAD THEN FILL IN _PLN2IPU TO _PLNnIPU.

0.137 _PLNXIPU = PLANNED NON-PRIMARY WORKLOAD CPU TIME IN MSECS. SAME AS ABOVE BUT FOR NON-PRIMARY TRANSACTIONS ONLY.

ATTACHMENT A

```
*************************************************************************;
* PROGRAM TO COMPARE THE FORECASTED VERSUS ACTUAL IPUS ;
*************************************************************************;

MACRO _PLANUNI 92880  %          MACRO _ACT1IPU 0.03918 %
MACRO _PLANIPU 3188.9 %          MACRO _ACT2IPU 0.0000  %
MACRO _ACTLUNI 92693  %          MACRO _ACTXIPU 0.01515 %
MACRO _ACT1UNI 78652  %          MACRO _PLN1IPU 0.0379  %
MACRO _ACT2UNI   0.000  %        MACRO _PLN2IPU 0.0000  %
                                 MACRO _PLNXIPU   0.0137  %

DATA VARIANCE;

*** CALCULATE ACTUAL TOTAL NON PRIMARY WORKLOAD TRANSACTION VOLUME.;

 ACTXUNI = _ACTLUNI - (_ACT1UNI + _ACT2UNI + _ACT3UNI);

*** CALCULATE ACTUAL IPU BASED ON ACTUAL WORKLOAD VOLUMES MULTIPLIED
  BY PLANNED CPU TIME PER TRANSACTION. NEEDED FOR MIXVAR CALCULATION.;

 ACTLIPU = (_ACT1UNI * _PLN1IPU) + (_ACT2UNI * _PLN2IPU) +
           (_ACTXUNI * _PLNXIPU) ;

*** NEXT THREE CALCULATIONS ARE BASED ON EQUATIONS DEPICTED IN ANALYSIS OF
  VARIANCE OF WORKLOADS, BY DAVID VINCENT, OCT 1986 EDP PERFORMANCE REVIEW;

 VOLVAR = (_PLANIPU / _PLANUNI) * (_ACTLUNI - _PLANUNI);
 MIXVAR = ((ACTLIPU / _ACTLUNI) - (_PLANIPU / _PLANUNI)) * _ACTLUNI;
 RATEVAR= ((_ACT1IPU - _PLN1IPU) * _ACT1UNI +
           (_ACT2IPU - _PLN2IPU) * _ACT2UNI +
           (_ACTXIPU - _PLNXIPU) * ACTXUNI);

DATA TOTAL;     SET VARIANCE;
 PLANIPU = _PLANIPU;
 TOTLIPU = (_ACT1IPU * _ACT1UNI) + (_ACT2IPU * _ACT2UNI) +
           (_ACTXIPU * _ACTXUNI);
 TOTVAR    = VOLVAR + MIXVAR + RATEVAR;
 PCTVOL    = VOLVAR / TOTVAR * 100;
 PCTMIX    = MIXVAR / TOTVAR * 100;
 PCTRATE   = RATEVAR / TOTVAR * 100;
 PCTOTVAR = TOTVAR / _PLANIPU * 100;
 PCTOTVOL = PCTVOL * PCTOTVAR / 100;
 PCTOTMIX = PCTMIX * PCTOTVAR / 100;
 PCTOTRAT = PCTRATE * PCTOTVAR / 100;
 FORMAT PCTVOL PCTMIX PCTRATE PCTOTVAR PCTOTVOL PCTOTMIX PCTOTRAT PERCENT.;
 PROC PRINT;   TITLE 'VARIANCE FACTORS';

DATA PRINT;     SET TOTAL ;
 FILE PRINT HEADER=H NOTITLES;
 PUT /@20 'PLANNED INFORMATION PLANNING UNITS FORECAST FOR A'/
          @20 'REPRESENTATIVE PEAK HOUR: TOTAL CPU TIME IN SECONDS ='
          @73  PLANIPU 7.1
        //@20 'ACTUAL INFORMATION PLANNING UNITS CONSUMED FOR A'/
          @20 'REPRESENTATIVE PEAK HOUR: TOTAL CPU TIME IN SECONDS ='
          @73  TOTLIPU 7.1
        //@20 'PERCENTAGE OF VARIANCE DUE TO  ' /
          @20 'TRANSACTION VOLUME.....' @52 PCTOTVOL 5.1//
          @20 'PERCENTAGE OF VARIANCE DUE     ' /
          @20 'TO TRANSACTION MIX .....' @52 PCTOTMIX 5.1//
          @20 'PERCENTAGE OF VARIANCE DUE TO  ' /
          @20 'CPU USAGE OR RATE PER          ' /
          @20 'TRANSACTION .....' @52 PCTOTRAT 5.1/
          @52 '------' /
          @20 'TOTAL VARIANCE .....' @52 PCTOTVAR 5.1///
          @20 '* A negative variance indicates the actual results were';
          @24 '   under the forecast';
 RETURN;
 H: PUT // @35 'WORKLOAD VARIANCE ANALYSIS' /
          @30 'ACTUAL VERSUS PLANNED IMS CPU USAGE' /
          @31 'IMS SUBSYSTEM APRIL 1987' //;
 RETURN;
```

*American Express makes no warranty either expressed or implied, as to the accuracy or usability of the program.

ATTACHMENT B

Disk Cache Performance Modeling

Michael A. Salsburg
UNISYS

ABSTRACT

In order to reduce the performance bottleneck caused by physical disk accesses, a large global disk cache has been implemented. During the design phase of this product, I/O workloads were studied and modeled. This paper describes the disk I/O subsystem, the I/O model and the results of the modeling, emphasizing the benefits of this style of disk caching.

Introduction

The end-user of a computer experiences the effects of computer performance as the delay between commands and responses. The delay between these events can be broken into transmission delay, processor delay and I/O access delay. As processor speeds increase, and line costs decrease, the I/O access delay is becoming a dominant factor in computer performance evaluation.

This paper presents a model of the I/O subsystem for the UNISYS A series systems. Since little has been published in this area, a description of the I/O subsystem's components is included. A trace-driven simulation model has been developed to observe the effects of various performance parameters using customer workload traces. A new feature, the Software Disk Cache Module, has been modeled and studied over the past two years. It significantly reduces the I/O bottleneck. Results from experiments in both modeling and real systems are included.

1. Workload - The Nature of Disk I/Os

Disk I/Os are requested by application processes on the host. Typically, these processes wait until the I/O is completed before continuing. As processor speeds continue to increase, the actual elapsed time, or "wall clock" time, for an application depends more heavily on the mechanical delays that are inherent in the disk subsystem. The events that contribute to the delay are effectively the same for disk READs (transfer from the spindle to main memory) and WRITEs (transfer from main memory to the spindle).

The arrival of these I/O requests is the workload of the disk subsystem. Traditionally, it has been assumed that these arrivals are a Poisson process. This assumption is used for two reasons. First, it reflects a high degree of conformity to actual arrival patterns when many independent processes are requesting service from a finite number of disks. The second reason is the tractability of mathematical models. Figure 1 shows the percent of I/O processing that was offered for a specific disk in one of the typical workloads studied.

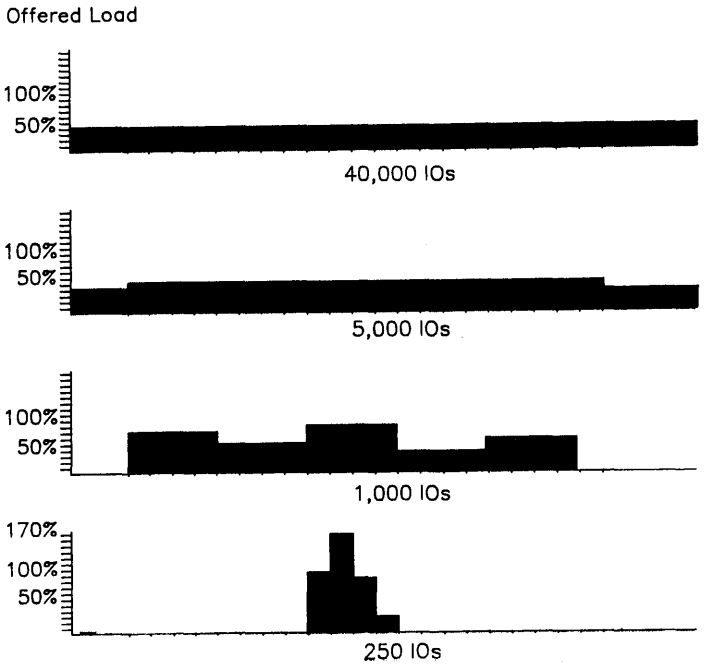


Figure 1

Chart (a) shows that, for a 40,000 I/O batch, about 40% of the capacity of the unit was requested. Chart (b) shows the offered load for a 5,000 I/O subset of that batch. Inspecting this batch separately shows that about 50% of the unit's capacity was requested, while other batches were lower. This batch is again subdivided in chart (c). Here, the offered load is analyzed as 1,000 I/O batches. In chart (d), the intervals of 250 I/O batches are displayed. Notice that one of the batches indicate an offered load of over 170% of the unit's capacity. Requests of over 100% indicates queueing between batches.

This level of variability is indicative of the Poisson process. There are two properties of I/O requests that do not fit the model of the Poisson process. First, I/O requests generally appear as bulk arrivals. In fact, the variability of actual workloads appears to be much higher than the Poisson process. Secondly, the arrivals are not independent. The 170% offered load was primarily generated from one application process. The determination of a more realistic description of the arrival process of I/Os may reveal better and more accurate modeling in the future.

2. The I/O Subsystem

This section will describe the events that occur in processing an I/O request. It includes a description of the disk caching that is used to reduce the delay an application process encounters when an I/O request is made.

Figure 2 shows where the Input Output Subsystem Module (IOSM) is positioned in the I/O process. Section 2.2 contains a discussion of the Cache Submodel. Section 2.1 discusses the IOSM submodel.

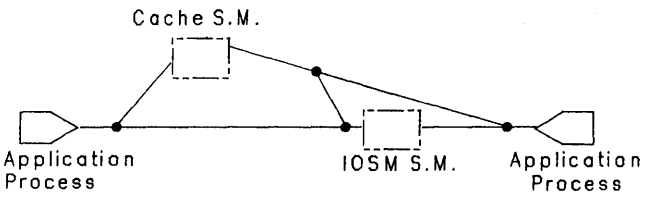


Figure 2

2.1 Standard (non Cached) I/O Request

Figure 3 displays a simple picture of the key elements in the I/O subsystem. In UNISYS A series machines, an I/O request is sent through the Message Level Interface (MLI) to the Data Link Processor (DLP) to the unit controller and finally to the unit. Many DLPs are connected to the host on a single MLI. Many units may be connected to a host through a single DLP and controller. For all practical purposes, the DLP and controller can be thought of as a single entity, since one does not process without the other. The two names will be used interchangeably.

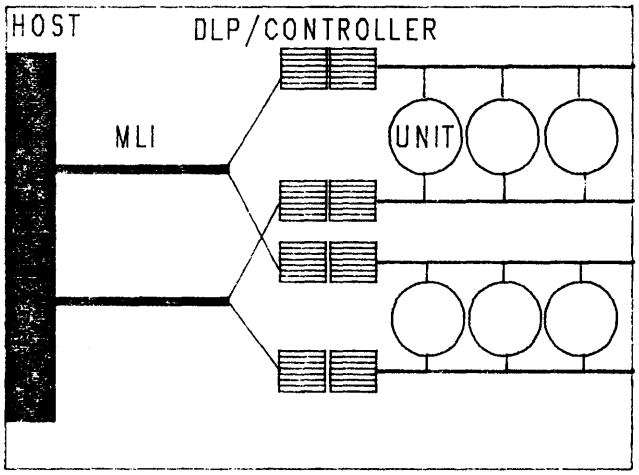


Figure 3

Figure 4 presents the detail of the IOSM submodel. The flow is from left to right. It is assumed in this example that a READ request has been issued to a unit. When an I/O for a particular unit becomes active, all future I/Os for the active unit are queued until that I/O completes.

The first event in the submodel is the acquisition of the Unit in the Unit Submodel (S.M.). Next, the command for the I/O must be sent through the MLI to the controller via the DLP. Once the command enques in the DLP, it is processed for **D1** milliseconds. If there is a need to move the unit's arm to the requested cylinder, a seek delay occurs for **D2** milliseconds. Once the seek delay completes, the controller is acquired again. The controller is not released until the final data

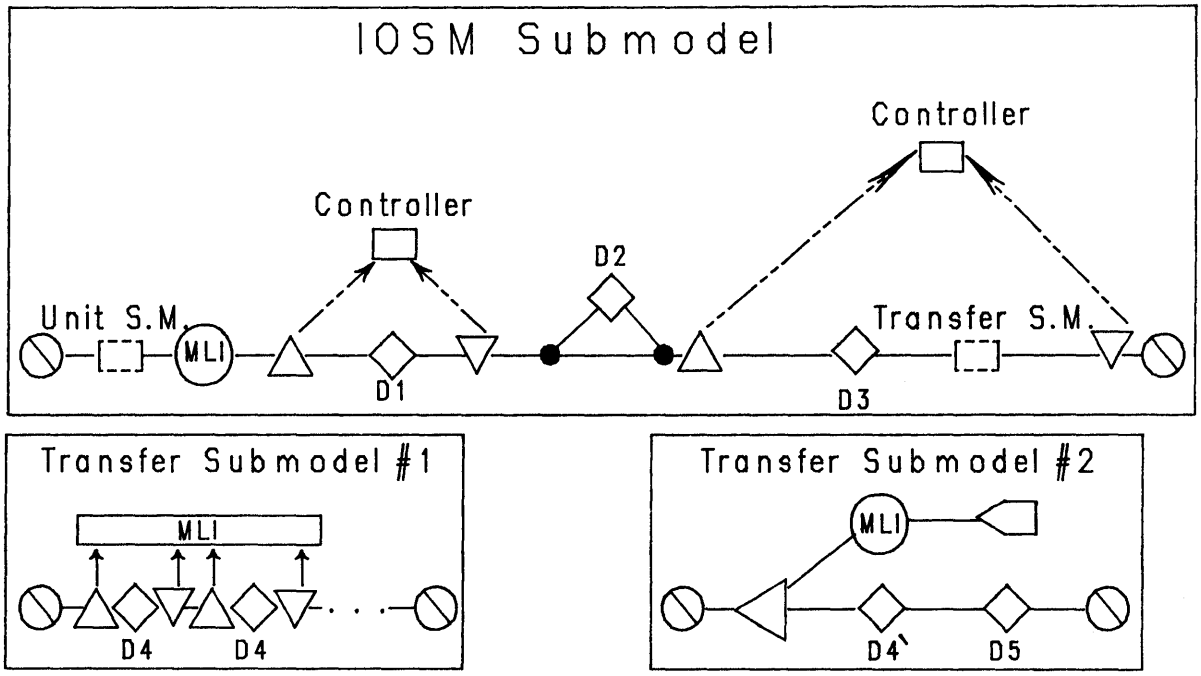


Figure 4

transfer completes. During this period, there is a delay for additional controller time and latency (D3). Once the data revolves under the READ/WRITE head, the transfer of data occurs. Two submodels are shown for this transfer. They will be discussed in section 4.1. The I/O completes when the data is successfully transferred.

Notice that the controller is busy during the latency phase. IBM compatible systems use a different scheme, called Rotational Position Sensing (RPS). See [2],[3] for a discussion of RPS modeling.

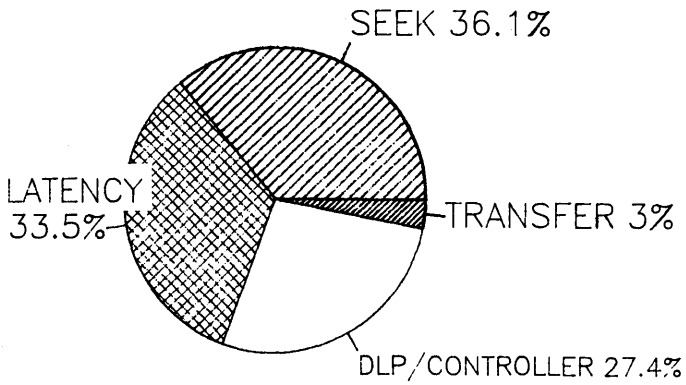


Figure 5

Figure 5 compares the amount of delay in seek time, latency, transfer and DLP/Controller processing for a thin film disk (Memorex 3680). Notice that the majority of the delay is devoted to setting up the transfer, while the transfer time is relatively negligible (3%). Disk caching has been introduced to reduce most of these delays.

2.2 Cache I/O Requests

A large software disk cache has been implemented for A12 and A15 UNISYS systems. Disk caching has been used successfully in the past to reduce I/O delays. The memory for software disk cache resides in the host's system memory. Many of the current disk caches have been implemented by introducing memory into the controllers and caching at the controller level. By using main memory, all disk units connected to the host may share the same cache. The size of the cache may be quite large (144 MBytes). The following is a brief description of the disk caching scheme.

In Figure 2, the placement of the disk cache submodel is shown. If a unit has been assigned the caching attribute, each I/O for that unit is

inspected by the cache algorithm before it enters the unit queue. When a READ request is found, the address of the disk data is inspected. If this address is not currently in the cache (cache 'miss'), the I/O command is replaced by a request for the entire track of disk in which the data resides. This request is then passed to the IOSM. The full track is read into the cache and tables are updated accordingly. If a READ request arrives for data already in cache (a cache 'hit'), the data is transferred from the cache memory to the application's I/O buffer without requiring IOSM resources.

For WRITE requests, all WRITEs are written to the physical medium. If a WRITE request is made for data in cache, the cache data is updated in addition to the physical update.

Memory is managed using a least recently used (LRU) algorithm. When memory is needed to hold a new disk track, the least recently used track is discarded. For a full description of caching and various algorithms, refer to [4]. In the terminology of Smith's paper, the cache algorithm is a write-through, write-update cache.

The performance issues of caching involve the effects of full-track transfers on the system, the trade-off of slower full-track READs (cache miss) for faster memory-to-memory transfers (cache hits) and the effects of cache size. These will all be presented in later sections. The next section is used to present the model that was developed to explore these issues.

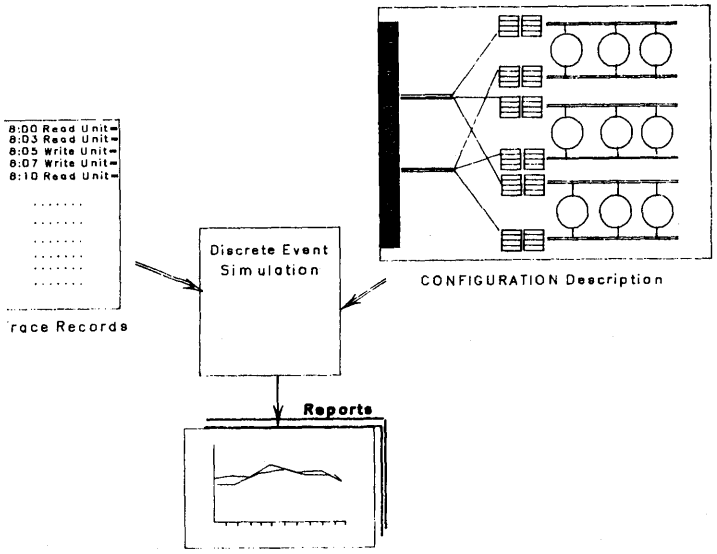


Figure 6

3. The IOSM Model

The performance of the I/O subsystem was studied using a trace-driven discrete event simulation model. It should be mentioned that, although many simulations involve Monte-Carlo methods, the majority of the events were deterministic. The simulator was used to evaluate the complex distributions and interplay that is difficult to predict using mathematical modeling.

Figure 6 presents a simple diagram of the simulation system. Traces of customer workloads were used to generate the arrival times and nature of the disk I/Os. A configuration file was used to describe the connectivity of the system. In this way, the simulation could be easily rerun and compared using alternate configurations. For seek I/Os, a mathematical model was used to produce seek delay times. The latency was calculated directly using the position of the data and the simulation time. DLP/controller delay times were determined from benchmarks. I/O performance metrics are calculated for batches of I/Os. Although the batch size is a parameter, the size was kept at 1000 I/O batches for consistency throughout the project. Autocorrelation of consecutive I/O delays is negligible beyond groups of 20.

Validation was an iterative project. As understanding of the system increased, re-validation helped identify key parameters of the system. The measured elapsed times for the I/Os were included in the trace data. One goal of the modeling project was to discover the minimum parameters necessary to accurately account I/O delay. The model appears to generate similar delays when compared to the workload traces. For non-seek I/Os, the average error was 3% of the average delay. For seeks the error was 4% of the actual average delay.

The caching algorithms were implemented using library calls from the simulator. The library manages the LRU data structures. The consistency of the LRU algorithm of the model is guaranteed to reflect the actual cache algorithm because the implementors of the software and the modelers shared the same algorithms.

4. Model Results

Using the workload description, the basic components of the I/O subsystem and the model, many scenarios were examined. The following subssections provide results from a few of the experiments.

4.1 MLI Modeling

In developing a simulation model, the question of granularity must be addressed. If the model accounts for short, frequent events (on the level of microseconds), while other less frequent events occur at a higher granularity, processor requirements are unreasonably high. The transfer of full track data across the MLI needed to be studied. It was decided to implement the actual buffering and bursting of data across the 8 megabyte/second MLI. Figure 4 shows the Transfer submodel #1. Since many DLPs share the MLI, the DLP connects and disconnects many times to transfer the data. Queueing will occur when a connection is requested; however data is being transferred from a real-time, spinning disk. If the queue size is too high, the unit is forced to disconnect from the DLP and revolve back to the remainder of the data (disk slip). This disk slip adds an additional 16.67 milliseconds to the I/O delay.

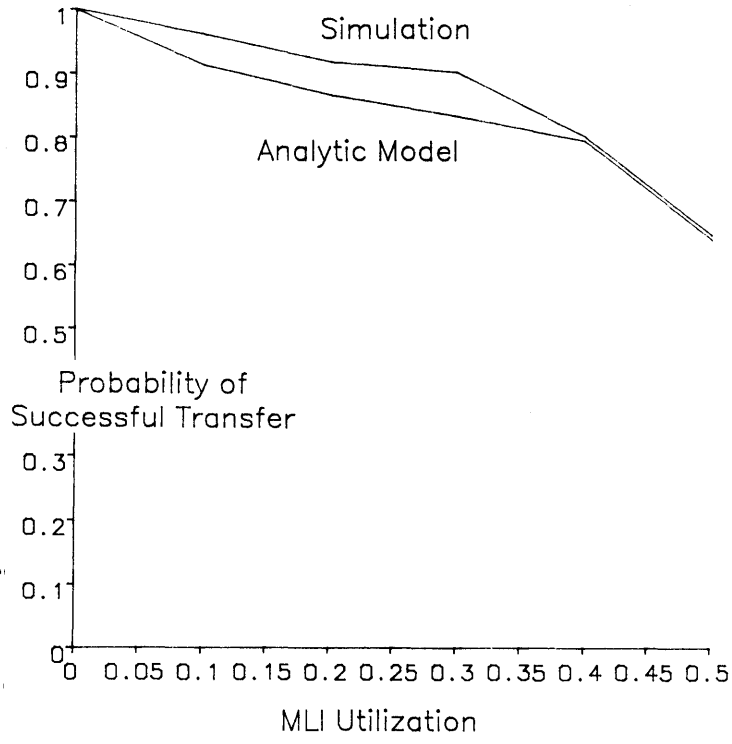


Figure 7

A mathematical model of the probability of disk slip was developed and compared to the simulation. The purpose was to replace the simulation events of MLI transfer by a simple delay. In Figure 4, Transfer Submodel #2, The multiple connections and delays are replaced by a single delay **D4** with an additional, calculated delay for disk slips. This alternate submodel reduced the processor requirements by over an order of magnitude.

Figure 7 resents the validation of this approach. Notice that at 50% MLI utilization, 35% of all I/Os will have an additional delay of 16.67 milliseconds. In all extant systems, the MLI utilization is usually under 0.8 megabytes/second. A problem will occur if a Line Expansion Module (LEM) is used. This module is often used to cross-connect DLPs for redundancy. Unfortunately, the LEM reduces the effective bandpass of the MLI by one-half, which means the utilization is doubled on these systems. MLI utilization must be carefully considered when cached disks are connected using LEMS.

The analytic formula for predicting the probability of a disk slip is:

$$P(\text{disk slip}) = 1-(1-p^{n})^{m}$$

The device utilization is **p**. The power of **n** is the minimum number of buffers in service and queued that will cause a disk slip (assumed to be 5.5 in this study). The power of **m** is the total number of times the DLP will attempt connection to the MLI to transfer a full track of data (assumed to be 20).

Due to the accuracy of this analytic model, a portion of the simulation could be replaced by simple arithmetic operations. This reduction made the modeling of the entire IOSM feasible.

4.2 Profile of Workloads under Test

The remainder of this section will discuss various aspects of disk cache performance. Throughout this discussion, a single workload will be cited. It would be preferrable to display multiple examples, but there is a limit to the size of this paper. The workload is a trace of a customer sites'

online system. Whenever performance is discussed, the usual caveat of "your mileage may vary" must be emphasized. This section is also intended to present the small number of parameters that need to be considered in evaluating any workload.

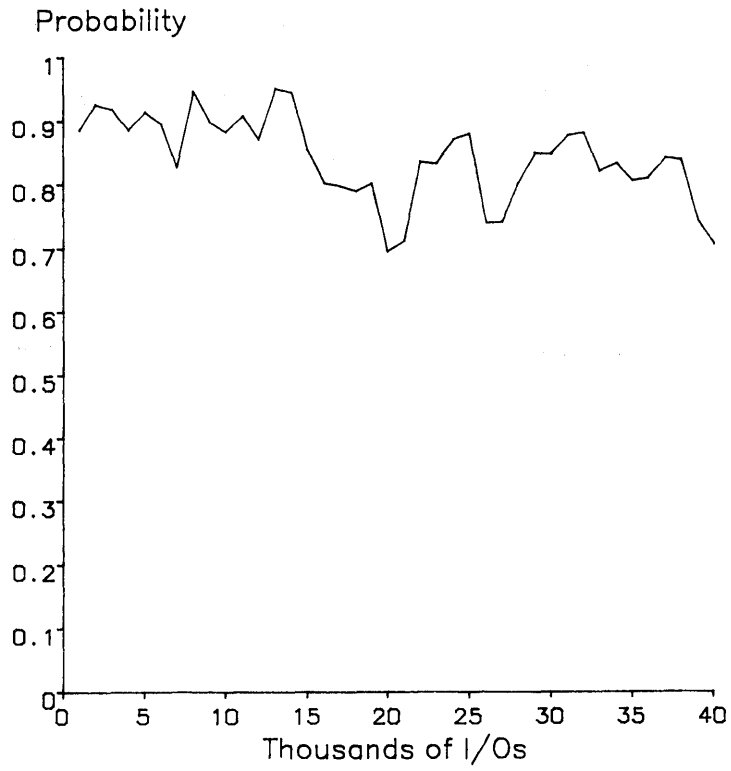


Figure 8

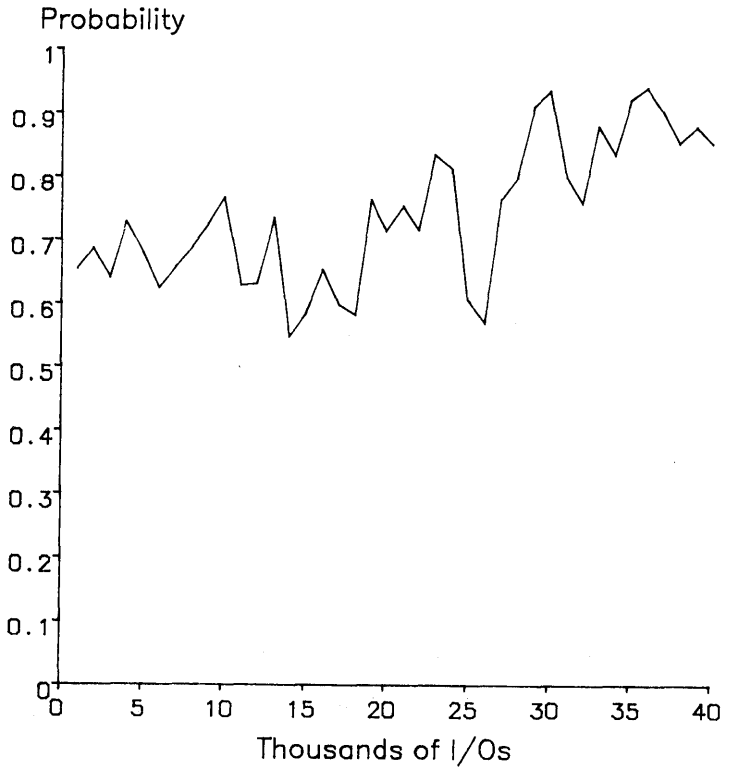


Figure 9

4.3 Parameters of Cache Performance

Figures 8 and 9 present the most important characteristics effecting cache performance. In Figure 8, each batch of 1,000 I/Os was assigned a probability of having a READ request. The other metric that is necessary is the probability of the READ being a cache 'hit'. This evaluation requires processing the I/O trace through the LRU algorithm. The probability is affected by the cache size. The smaller the cache size, the less tracks in cache and the lower the probability of finding the data resident. Figure 9 shows the probability of a READ 'hit' given a cache size of 100 megabytes.

The basic philosophy of global disk cache is that the caching algorithms need not be too elaborate if the memory size is sufficient. Since system memory is capable of growing to 192 megabytes on an A15 system, 100 megabyte caches do not seem unreasonable. Figure 10 shows how the overall cache hit probability changes as a function of memory size. This graph is typically displayed using a logarithmic scale for memory size, which displays a steeper curve. However, most people involved in the cost of memory do not consider it logarithmic, therefore, the linear scale is used.

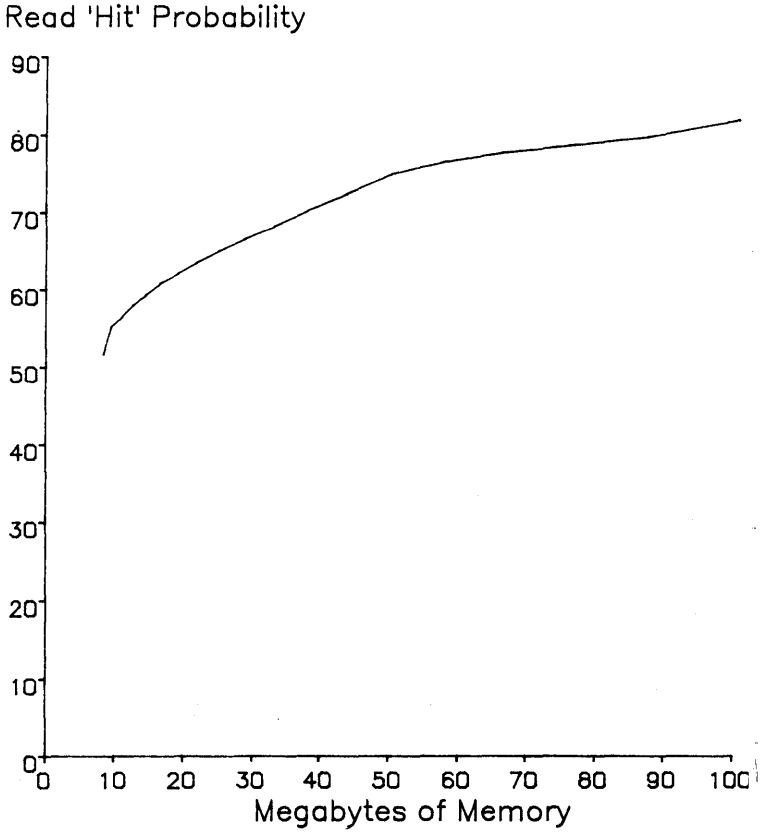


Figure 10

4.4 I/O Service Improvement

Many workload traces were tested on the cache simulation model. One method of comparing cached vs. non cached performance is by the reduction of I/O service time. The I/O service time is defined as the elapsed time that the I/O is active. From the application process view, it is the I/O delay minus the unit queueing delay. The cache improvement (CI) is the ratio of the non-cached service time divided by the cached service time. As the probability of READs increase and/or the probability of cache 'hit's increases, the cache improvement metric increases. Figure 11 shows the cache improvement metric calculated for each 1,000 I/Os of the workload. Workloads with consistent 90% cache hit and READ probabilities show I/O delay reduction to one-fifth of the non cached delay.

Notice that the improvement does not go below one. This would indicate that the cache was slowing down the I/O delays. Adverse effects of controller cache has been reported in the literature. In order for controller caches to remain effective, the rule-of-thumb is to keep the READ probability and the READ 'hit' probability above 70%. If these rules are not followed, the excessive controller utilization, due to cache management, will cause delays which will elongate the I/O delay times for all units connected to the controller [1]. The cache is extremely resilient to these factors. It will be shown that a low probability of READs may have no adverse effect on cache performance. Furthermore, READ 'hit' probabilities can drop below 50% without causing I/O performance degradation.

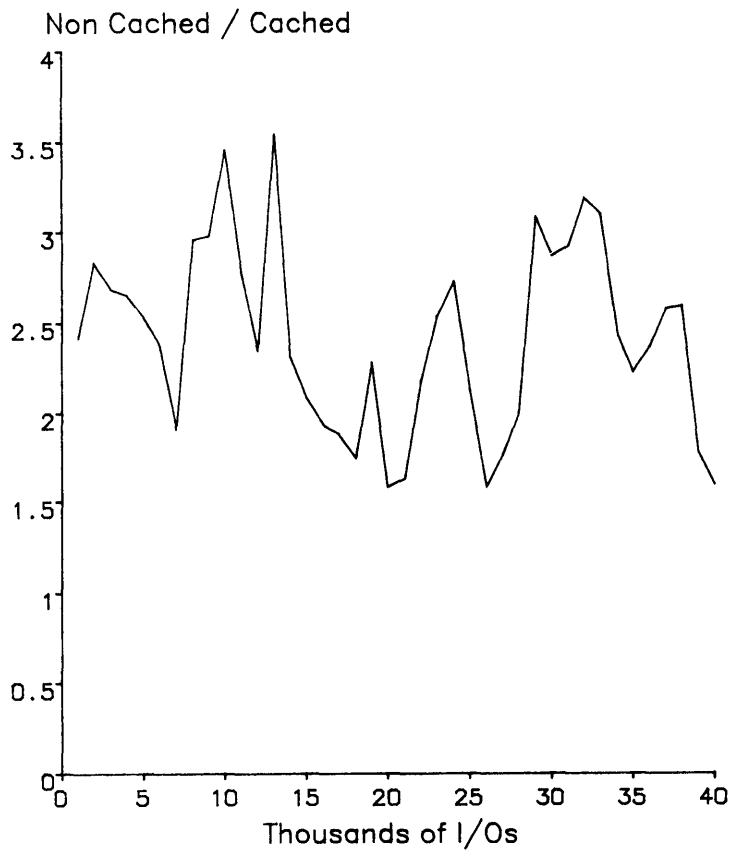


Figure 11

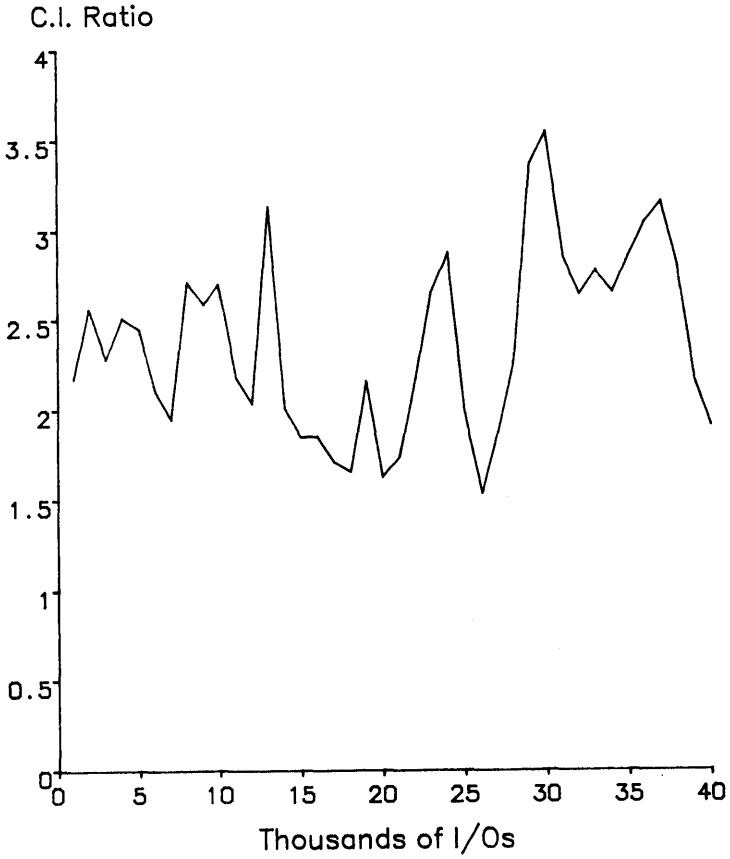


Figure 12

4.5 A Model of Cache Improvement

It would be extremely useful to have a simple algorithm to compute the cache improvement, without the necessary simulation requirements. Along with the original research into global memory disk cache schemes, Tim Fender of UNISYS developed a simple formula for predicting the cache improvement ratio:

$$CI = 1/(C_1 + 1 - P_R(1 - C_2 * (1 - P_H)))$$

where:

P_R = probability of READs
P_H = probability of READ 'hit'
C_1 = average increase of processor time
C_2 = average ratio of service for full track I/O vs "regular" I/O

Figure 12 shows the CI as calculated with the above formula. The constant C_1 represents the additional CPU resources necessary to process cache misses, averaged over all of the I/Os. For this workload, .02 (milliseconds) was used. It should be noted that, when a cache 'hit' occurs, the processor utilization is less than that of an ordinary READ I/O. The constant C_2 is used to represent the additional delay for a full track transfer over the delay for an 'average' I/O. Since the transfer time is not the majority of the I/O delay, and the average transfer size was 2800 bytes, C_2 is set at 1.5. The formula underestimates the actual cache improvement; this is due to the controller queueing delay improvement. Since the cache has reduced the utilization of the controllers, the controller queueing has been reduced, thus lowering the delay time for an active I/O. The effects and distribution of controller queueing are now under study.

Using this lower estimate of cache improvement, examine the formula for $\%_R = 0$. If the system encounters all WRITE I/Os, the overall cache improvement is

$$1 / (1 + C_1).$$

For C_1 = .02, this implies a degradation of the I/O delay by 2%, due to additional processor overhead. Assuming there are READ I/Os, a degradation will not occur if

$$C_1 * (1 - P_H) < 1.$$

For C_2 = 1.5, the P_H must be above 1/3.

4.6 Elapsed Time Improvement

Although the CI metric is easy to calculate and considered a reliable indicator, the final success rests in the actual difference in elapsed times encountered when running application processes. This difference in elapsed times includes the reduction in unit queueing. Notice that WRITEs and READ 'misses' are now being processed in a system that is less utilized. Therefore, the overall unit queueing is reduced. In this way, all I/Os experience less delay if cache 'hits' are occurring. Figure 13 compares the total elapsed time for the workload. Notice the scale for Elapsed Improvement. This is the ratio of non cached I/O delay over the cached I/O delay.

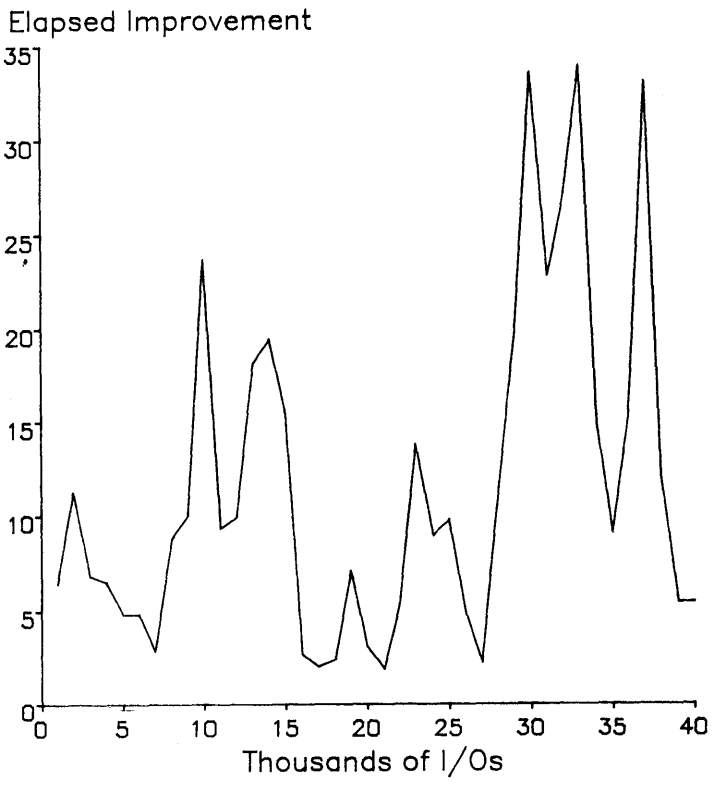


Figure 13

5.0 The Product

All of the modeling results were completed before the product was implemented. The first test site for the software was an internal system . The experience of using the product met the expectations predicted by the modeling. The CI metric has been tested with actual elapsed times reported for applications that were run with and without disk cache. Some extraordinary results were reported recently where a production system ran an overnight batch program in three of the usual eight elapsed hours. At one point, two units were each logically handling over 1500 I/Os per second. On the other hand, a site consisting of totally random update transactions showed a decrease in I/O delay ranging from 8 to 30%.

6.0 Conclusion

The nature of the UNISYS A series I/O subsystem has been presented, emphasizing the performance of the global disk cache feature. As has been shown, the benefit of the caching feature on the I/O bottleneck may vary greatly from workload to workload. Ultimately, some analysis is necessary to insure that the improvement will justify the cost. Hopefully, this paper will bring to light the main issues that are involved with this sort of analysis.

The modeling project has raised many interesting questions, such as the real nature of I/O arrivals, and the distribution of various delays throughout the subsystem. A four month benchmarking project has just been completed. The results of these experiments are the subject of a future paper. The understanding and configuration of large disk subsystems grows in importance as processor times continue to shrink.

6.0 Acknowledgments

I would like to emphasize that this work was built on the strong foundation of Tim Fender's research and preliminary modeling. I would like to thank him for his encouragement and patience. Technical assistance was provided by the principal software disk cache implementor, Will Stratton.

Bibliography

[1] Kovach, R. P, **DASD Cache Controllers: Performance Expectations and Measurement,** Proceedings of CMG '86.

[2] Salsburg, M. A, **A Statistical Approach to Computer Performance Modeling**, ACM Sigmetrics 1987.

[3] Salsburg, M. A, **Examining RPS Delays,** Transactions of CMG, Winter 1987.

[4] Smith, A.J, **Disk Cache-Miss Ratio Analysis and Design Considerations**, ACM Transactions on Computer Systems 3, 3, August 1985.

A Method For Reporting Cashed I/O Subsystem Performance

Nancy Nearing
The Washington Consulting Group

ABSTRACT

Recently, at a client site, we developed the capacity to monitor cached I/O subsystem performance for more than two hundred DASD volumes capable of being cached. The volumes were configured on approximately 35 strings with NAS model 23 cache controllers, shared among four Amdahl 5870 attached processors, three of which were running MVS/XA, and one of which was running MVS/370. Each string usually contained at least one cached volume, but the configuration was prone to frequent changes.

Using the SMF 244 and 245 records produced by the RMF cache reporter, combined with standard RMF data on response time and start I/O requests, we developed a series of SAS programs to depict cache performance characteristics for the entire configuration. These reports show the additional connect time, caused by cache activity, which is invisible to RMF. This additional connect time is a symptom of the increased controller utilization caused by staging operations. It can cause increased RPS reconnect delays and overall I/O performance degradation; therefore, it should be watched carefully.

1. Background:

The basic measure of cached DASD performance is response time. If cached volumes show significant response time improvements over non-cached volumes, then cache is operating satisfactorily, and there is no need for tuning [1]. However, if the response time improvement is not as great as anticipated, some adjustment may be required. In order to adjust, or tune, the cached I/O subsystem, one needs to consolidate the "internal" cache management data with response time data.

Consolidated cache performance data, however, is difficult to come by. Since cache management functions are "invisible" to host computer software monitors like RMF and SMF, the performance analyst has no easy way of measuring cached DASD performance. The IBM 3380 storage director maintains cache management data which is used by the IBM Cache RMF Reporter to produce a voluminous, detailed report of cache performance indicators, but response time is not one of them. The report could be compared to a response time report to provide a complete picture of performance.

This is a cumbersome process at best; witness the discussion of cache controller tuning in [4]. The Cache RMF Reporter was used to provide cache management data, Omegamon was used on-line to check response times, and the RMF DASD Activity Report was consulted for volume activity information. This was all for tuning a single controller. The process could only become more awkward as more cached controllers are installed and need to be tuned.

There is a better way. In order to produce its reports, the Cache reporter writes the 3880 counts out to SMF user records. This paper describes how we merged the cache management data contained in the SMF user records with device activity data provided by RMF. Once the data was merged, we produced a consolidated, summarized report that is currently used for tuning an DASD installation with more than 30 cached controllers serving over 350 DASD volumes.

This paper describes our methodology for creating consolidated cache management data and producing summarized performance reports at a large site. Although we completed the project in the context of a commercial performance data base, the techniques we used and insights we gained are applicable to other situations and environments.

This paper is organized in five sections, a brief background to the problem (this section), an introduction to cache management data and terminology (which may be skipped by those familiar with the subject), our methodology for merging the data and creating the reports, and a thumbnail sketch of how to tune a cached system using the reports.

2. A General Description of Caching.

The primary objective of cached DASD is to improve I/O response time compared to conventional I/O. The cache is RAM shared by storage directors which have been modified to query the cache before initiating a disk access to retrieve a record. If a read request references a record currently located in the cache, transfer of data from the cache to the host can begin immediately, and proceed without the delays associated with conventional disk access [3].

A read request for a record located in the cache is referred to as a read hit, or a cache hit. If the record is not located in the cache, and a conventional I/O operation must be initiated, it is referred to as a read miss. Since write operations inevitably have to complete a conventional I/O, they do not benefit from caching. Therefore, write operations and read misses are collectively referred to as cache misses. [1] estimates read miss service time (which should be approximately the same as service time to a non-cached 3380) as 20.4 ms, and read hit service time as 2.8 ms - a savings of more than 17 ms.

Although a read miss does not take significantly longer to complete than a non-cached read operation, once the request is completed, the cache manager will "stage" data from disk to the cache, in hopes of preventing future misses. The staging operation ties up the device, channel, and control unit for the duration of the operation. Therefore, excessive staging causes cached controllers to be significantly busier than conventional disk controllers. The extra burden placed on cached controllers by staging activity is referred to as "cache overhead". This overhead is the biggest performance liability of caching and will produce significant RPS reconnect delays on cache misses. A saturated cache controller will cause overall subsystem performance to degrade when I/Os that must bypass the cache and perform conventional disk I/O operations encounter excessive reconnect delays. [3]

Aside from response time, which provides an indication of benefits received from caching, the most important indicator of cache performance is the amount of "invisible" controller utilization caused by staging, which can provide an indication of liabilities caused by cache. Response time data for a volume or a string can be derived from RMF. The only way to estimate how much additional con-

troller utilization is being caused by staging is to know how much offline staging activity there is. For a 3880 model 23, this data can be derived from the counts maintained by the storage controller itself.

Other major indicators of cache performance are hit percent, read hit percent, and hits per stage ratio. These measurements can all be derived from the cache management data. Hit percent is a measure of the number of operations that benefit from the cache compared to the total number of operations requested -- in other words, read hits compared to total I/O operations. Read hit percent is the comparison of read hits to total read requests. The hits per stage ratio compares the total number of read hits for the measurement interval with the total number of stages that took place. This provides an indication of how many read requests can be satisfied by each staging operation. the larger the number, the more benefit is being derived from the cache. As we noted above, when performance problems occur in a cached subsystem, they are usually associated with the extra controller load in cache due to staging activity following read misses. The hits/stage value provides a useful index of overall cache improvement vs. overhead [3].

The best guidelines for tuning a cached subsystem are obtained by measuring its performance under a variety of conditions, and noting the circumstances under which it tends to degrade. However, when this type of information is not available, one may refer to general "rules of thumb" for guidelines. Published rules of thumb suggest that an appropriate workload for caching should have at least a 70% percentage of read hits, and a 2:1 read/write ratio. [1]

3. Methodology.

RMF reports response time data that is an accurate measure of cached I/O response time as seen from the host computer. However, because the cache management functions are handled off-line, and therefore are invisible to RMF, RMF data alone is virtually useless in tuning a cached subsystem. In order to provide a complete picture of the internal workings of the cached subsystem, information about the cache management functions must be combined with the RMF data. Data from any version of RMF will provide response time information; RMF operating under MVS/XA is especially useful since it provides an additional measure of connect time which can be used to derive an indication of total control unit utilization.

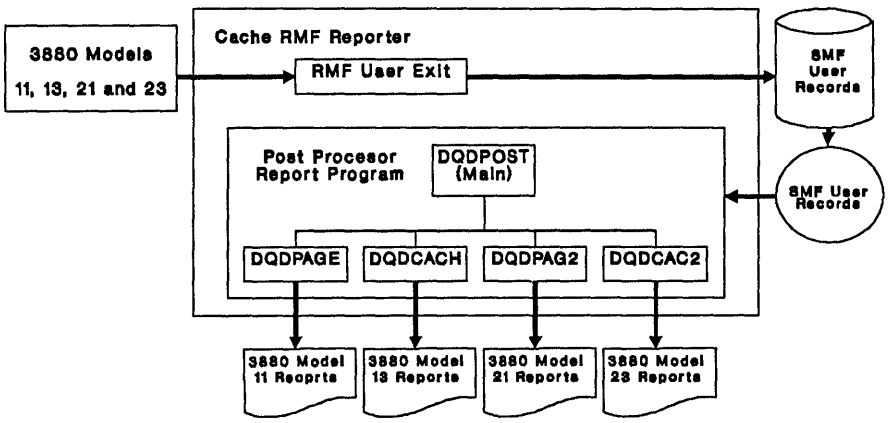
The IBM product, Cache RMF Reporter, provides information about the cache management functions. It consists of an RMF User Exit Routine, and a post-processor program. The exit routine is entered at the beginning of each RMF Monitor I session, and at the end of each RMF measurement interval [2]. Each time it runs, it issues the Access Method Services (IDCAMS) LISTDATA command, which uses the Sense subsystem Counts CCW to obtain the device counters and subsystem status for each storage controller. Since the LISTDATA command does not reset the counters, the user exit must then calculate the difference in the counts from the previous interval [1]. The results reflect all I/O activity seen by the controller over the measurement interval, and they, along with the subsystem status fields, are written to the System Management Facilities (SMF) dataset as user records. Note that the user exit does not calculate any ratios or percentages; these statistics are all derived by the post-processor program. The user exit will write a record of one type for each 3880 model 11 or 21 it encounters, and another for each 3880 model 13 or 23. The record types are specified by the user when the cache reporter is installed, and can be anything from type 128 to type 255, inclusive. The defaults are 244 for a model 11 or 21, and 245 for a model 13 or 23 [2]. The user record formats for the 3880 models 13 and 23 are included as an Appendix.

The post-processor program consists of a main routine, which reads the SMF records, and four subroutines, one for each 3880 model (11, 13, 21 and 23) recognized by the cache reporter. The subroutines calculate usage statistics, such as hit ratios and read/write ratios, from the records read by the main routine, and then generate the reports for each device [2]. The entire Cache RMF Reporter structure is illustrated in figure 1.

The Cache RMF Reporter post processor reports contain data solely from the storage controllers, and do not provide response time data as viewed from the host processor. In addition, since no data is provided on volumes brought up offline to the cache, the reports cannot show data for an entire subsystem unless all its volumes were cached over the reporting interval.

In order to provide complete information about the cached subsystems, we have substituted two SAS programs for the post- processor program. The first program is a daily SAS routine which calculates usage statistics, formats the data into two files. The first file

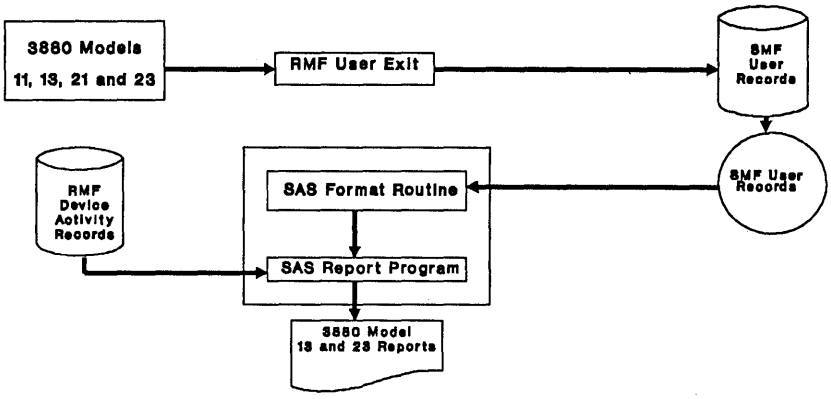
Figure 1. Overview of the Cache RMF Reporter



Adapted from the Cache RMF Reporter Program Description/Operation Manual

maintains data pertaining to an individual volume, and the second maintains data pertaining to an entire string. The second program is a report program, which merges the user record data by date, hour, and volume ID with RMF data on device activity, providing data for both cached and non-cached volumes. The cache reporting structure we are currently using is illustrated in figure 2.

Figure 2. Cache Reporting Strucutre Currently In Use



The data we report daily is a combination of RMF device activity data from all the MVS CPUs and data from the RMF Cache Reporter for one system only. The reason for using cache data from only one system is that the cache controllers are shared among all the MVS CPUs, and, since they are unable to separate the data they maintain by CPU, report their total statistics to each machine's RMF files. The IBM recommendation is to report management data from shared cache controllers to only one CPU. [2] At our site, data was reported to all the CPUs as a convenience; however, using cache data from all the CPUs would cause all the cache counts to be reported multiple times.

The use of both data sources enables us to produce reports of cache performance and overhead which include response times and I/O rates for both cached and non-cached volumes. Statistics for two full strings are shown on each page. We have found these reports to be extremely valuable for tuning because they provide complete cache management data, including response time and statistics for non-cached volumes, in a convenient summary form. Examples of our reports are shown in figures 3 and 4.

The following measurements and derivations are of particular interest: 1. SIO Rate; 2. Average Response Time; 3. Control Unit Data Transfer Rates; and 4. Control Unit Utilization invisible to RMF.

The SIO rate is the total number of I/Os reported for the period, divided by the period's duration in seconds, giving the number of I/Os per second for the reporting period. The source for the reported number of I/Os varies according to whether or not the volume was cached, and whether it was accessed by a processor running MVS/XA or MVS/370. In all instances, we report cache data if it is available. If it is not, we must make due with RMF data for comparison, even though it is not exactly equivalent to the cache data. If the volume was cached over the reporting period, the value reported is calculated from counts maintained by the controller. It is the sum of search read cache requests (RF7SRCR), write cache requests (RF7WCR), read sequential requests (RF7RSR), and write sequential requests (RF7WSR). If the volume is not cached for the reporting period, the total number of SIOs reported is the sum of the SSCs to that volume from the MVS/XA processors, and the SIOs from the MVS/370 processors.

Average response time is the average time it takes to complete an I/O operation, from the initial issue of the SIO or SSC command to the hardware interrupt indicating successful completion of the request. Average response time is calculated every interval for each processor's I/O activity to each volume. In order to combine this figure meaningfully across all four systems, we weight the recomputed average by the number of SIOs from each processor.

The information necessary to derive the cache data transfer rate is available only from the model 23 cached controller. According to [1], " ... the 3880-23 counts the number of bytes transferred on its various data paths. The bulk of the control unit utilization stems from data transfer operations, and is more related to the number of bytes transferred than to the number of I/O Requests executed. Hence the division of these counts can aid in approximating what portion of the control unit utilization is contributed by each of the following four types of operations:"

The four operations referred to are: cache to channel transfer, DASD to channel transfer, DASD to channel and cache transfer, and DASD to cache transfer. The bytes transferred from cache to channel are transferred without DASD access; they satisfy read hit requests. Those transferred from DASD to the channel satisfy I/O requests that specifically bypass the cache. The bytes transferred from DASD to cache and channel are the part of a read miss that satisfies the request for a record, and updates the contents of the cache at the same time. (Note that cache controllers prior to the 3880-23 did not transfer to cache and channel simultaneously. They first transferred the requested record across the channel, and then initiated a staging transfer to update the cache [5].) The bytes transferred from DASD to cache are bytes transferred in staging operations. They are the part of a read miss operation that causes caching overhead.

We also use the count of bytes transferred in staging operations to derive two measures of cache control unit utilization. The first measure is connect time, or the amount of time the controller is actually busy with staging transfers. The second is the percent utilization added by the caching functions.

Connect time in this case refers only to connect time incurred as a result of caching overhead, which is invisible to RMF on the host machine. It is calculated by dividing the total number of bytes transferred in staging operations by an approximation of the transfer rate per second. As discussed above, bytes transferred in staging operations are transferred from DASD to the cache (field RF7DCCDT in the SMF user record). The transfer rate for a 3880

DAILY CACHED DASD STATISTICS FOR THURSDAY, JANUARY 8, 1987

REPORTING PERIOD: 7:59:02 TO 15:59:09

CONTROL UNIT C60 - MODEL D23

STORAGE DIRECTORS 60 AND 63

CU	DEV	VOLUME	I/O RATE	RESP TIME (MS)	---READS PER SECOND--- REQ	HITS	HIT%	RD/WT RATIO	WRITE REQ/SEC	VOLUME'S SHARE OF $IO	VOLUME'S SHARE OF READ HITS	----BYTES TRANSFERRED PER SECOND----- --------DASD TO------- CACHE	CHANNEL	BOTH	---CACHE TO--- DASD	CHANNEL
C60	C60	B00033	0.13	20	.	.	.	0.0	.	0.21	0.00	.	.	.	.	.
	C61	B00034*	7.18	17	5.87	3.90	66.45	4.5	1.31	11.84	18.47	73590	0	14618	0	25949
	C62	B00037*	8.56	19	7.04	4.71	66.95	4.6	1.53	14.12	22.31	87067	0	17512	0	31945
	C63	B00039	0.06	16	.	.	.	0.0	.	0.10	0.00	.	.	.	.	.
	C64	B00025	0.32	24	.	.	.	0.0	.	0.54	0.00	.	.	.	.	.
	C65	B00027	0.23	23	.	.	.	0.0	.	0.38	0.00	.	.	.	.	.
	C66	B00029*	18.38	15	17.10	12.51	73.16	13.4	1.28	30.29	59.23	171856	0	32907	0	38547
	C67	B00031	7.00	30	.	.	.	0.0	.	11.55	0.00	.	.	.	.	.
	C68	A10124	10.45	53	.	.	.	0.0	.	17.23	0.00	.	.	.	.	.
	C69	A10188	0.01	1	.	.	.	0.0	.	0.01	0.00	.	.	.	.	.
	C6A	A10190	0.28	29	.	.	.	0.0	.	0.47	0.00	.	.	.	.	.
	C6B	B10045	0.14	22	.	.	.	0.0	.	0.23	0.00	.	.	.	.	.
	C6C	B10047	5.23	18	.	.	.	0.0	.	8.62	0.00	.	.	.	.	.
	C6D	C00002	1.36	25	.	.	.	0.0	.	2.25	0.00	.	.	.	.	.
	C6E	A00075	0.02	11	.	.	.	0.0	.	0.04	0.00	.	.	.	.	.
	C6F	A10192	1.30	29	.	.	.	0.0	.	2.15	0.00	.	.	.	.	.
CU	C60	TOTAL:	60.67	25	30.01	21.12	70.39	7.3	4.12	100.00	100.00	332513	0	65037	0	96441

FIGURE 4

DAILY CACHED DASD OVERHEAD STATISTICS FOR THURSDAY, JANUARY 8, 1987

REPORTING PERIOD: 7:59:02 TO 15:59:09

CONTROL UNIT C60 - MODEL D23

STORAGE DIRECTORS 60 AND 63

CU	DEV	VOLUME	I/O RATE	RESPONSE TIME(MS)	---READS PER SECOND--- REQ	HITS	HIT%	RD/WT RATIO	STAGES /SEC	READ HTS /STAGE	PERCNT OVRHD	CONNECT TIME	%CONTROLLER UTILIZATION	TOTAL HIT %
C60	C61	B00034	7.18	17	5.87	3.90	66.45	4.471	1.970	1.98	22.17	13:28.67	2.81	54.30
	C62	B00037	8.56	19	7.04	4.71	66.95	4.609	2.326	2.03	26.18	15:56.77	3.32	55.01
	C66	B00029	18.38	15	17.10	12.51	73.16	13.388	4.588	2.73	51.65	31:28.51	6.56	68.08
CU	C60	TOTAL:	34.12	16	30.01	21.12	70.39	7.288	8.884	6.73	100.00	60:53.95	12.68	61.90

model 23 is generally listed at 3MB per second; however, if we were to divide the bytes transferred by 3MB, we would arrive at the amount of connect time spent actually transferring data, with no allowance for the overhead associated with the channel protocol, etc. In order to derive a more realistic figure, we divide the bytes transferred by 2.5MB, which provides an allowance for connect time not devoted exclusively to data transfer.

The percent of time the controller was occupied with cache overhead functions is then determined by dividing the derived connect time (in seconds) by the duration of the reporting period (in seconds). Note that the percent utilization figure accounts only for control unit utilization attributable to caching overhead, and not for utilization reported to RMF. Under MVS/XA, it is possible to determine the control unit utilization reported to RMF by dividing the reported connect time by the reporting interval. The sum of control unit utilization for all processors accessing the string, plus that contributed by the caching overhead, would be equal to the total controller utilization for the configuration.

4. Using the Reports for Tuning.

According to Friedman [3], "If performance (of the cache) is not optimal, I/O response time will be no better than the baseline uncached subsystem, and in extreme cases, it can be significantly worse." Therefore, the first step in tuning the cache with the Cached DASD Statistics report, figure 3, is to look at the response times. If the average response time for the string and each of its volumes is acceptable, there is no need to look further.

If, however, cached I/O response time is not faster than uncached, one should examine the reports further. At this point the control unit data transfer rates should be examined. If a disproportionate amount of data is being staged (bytes transferred form DASD to Cache), the problem may lie in excessive controller utilization.

As mentioned earlier (section 2), the greatest performance liability of caching is the controller activity generated by staging. If the overhead report (figure 4) shows a high percentage of cache overhead controller utilization, the controller may be saturated. In this case, decreasing controller activity may substantially improve response times for the volumes on the string.

In order to decrease controller activity and improve response time, poor cache candidates should either be removed from the string, or excluded from the cache. The current school of thought is that removing a poor candidate from the string is a better solution than excluding it from the cache [3,4].

Poor cache candidates can be identified by a low read hit percentage, and a high overhead contribution. If however, all the volumes contributing to cache overhead are showing good read hit percentages and high SIO rates, poor response time may be caused by nothing more than too much activity. In this case, performance may be improved by replacing some of the most frequently accessed volumes with ones that are less in demand [3].

5. Conclusion

Most widely available cache performance reports seem to rely on only the data provided by the cached controllers, forcing the performance analyst to depend on a variety of sources for complete information when tuning a cached I/O subsystem. This paper has described a method whereby the cache data provided by the IBM Cache RMF Reporter may be combined with RMF response time and device activity data to provide a consolidated report which is far more convenient for the analyst to use. First, it discussed the background of the problem, then, some of the measurements which can be used in cache tuning, and the methodology for producing the reports. Finally, it provided a thumbnail sketch of how the reports may be used for tuning a subsystem.

APPENDIX:

SMF User Record Formats for the IBM 3880 Cache Controller, Models 13 and 23. Excerpted from [1].

REFERENCES:

[1] Guide to the IBM 3880 Storage Control Model 23 IBM, January 1985

[2] Cache RMF Reporter
Program Description/Operations Manual
IBM, November 1984

[3] Performance and Tuning of Cached I/O Subsystems
Mark Friedman
Proceedings of the Computer Measurement Group, 1984

[4] Degradation of IMS Performance Using 3880-23 Cache Controllers
C.S. Hanna
Proceedings of the Computer Measurement Group, 1986

[5] DASD Cache Controllers: Performance Expectations and Measurement
Roger P. Kovach
Proceedings of the Computer Measurement Group, 1986

BIOGRAPHICAL NOTE: The author, currently employed by the Washington Consulting Group, developed the capacity to monitor cached I/O subsystem performance while working with Wilson Hill Associates, Inc.

```
/*****************************************************************/
/*                                                               */
/*  FORMAT OF SMF RECORD FOR CACHING SUBSYSTEM STATISTICS        */
/*                                                               */
/*****************************************************************/
 DCL 1 SMFF5 BASED(SMFF5PTR) BDY(BYTE),/*                        */
                             /* CACHING SUBSYSTEM RECORD        */
    2 SMFF5LEN    FIXED(16), /* LENGTH OF RECORD                */
    2 *           FIXED(16), /* SEGMENT DESCRIPTOR              */
    2 SMFF5SYS    BIT(8),    /* SYSTEM INDICATOR                */
    2 SMFF5TYP    FIXED(8),  /* RECORD TYPE                     */
    2 SMFF5TME    FIXED(32), /* TIME SINCE MIDNIGHT IN .01      */
                             /* SECOND UNITS                    */
    2 SMFF5DTE    CHAR(4),   /* DATE                            */
      3 SMFF5YR   FIXED(16), /* YEAR                            */
      3 SMFF5DAY  FIXED(16), /* DAY                             */
    2 SMFF5SID    CHAR(4),   /* SYSTEM ID                       */
    2 *           FIXED(16)  /* RESERVED                        */
    2 SMFF5NDV    FIXED(16), /* NUMBER OF DEVICE ENTRIES        */
    2 SMFF5MDL    FIXED(8),  /* CACHING SUBSYSTEM MODEL         */
    2 *           FIXED(8),  /* RESERVED                        */
    2 SMFF5NSS    FIXED(16), /* NUMBER SUBSYS STATUS ENTRIES    */
      3 SMFF5RTN  FIXED(16), /* IDCSS01 RETURN CODE IF NDV=0    */
    2 SMFF5INT    FIXED(32), /* NUMBER SECONDS SINCE SUBSYSTEM  */
                             /* STATISTICS LAST REPORTED        */
    1 RF5SDATA BASED,        /* DATA PORTION                    */
/*****************************************************************/
/*                                                               */
/*  FORMAT OF DATA PROVIDED FOR CACHING SUBSYSTEM                */
/*                                                               */
/*****************************************************************/
      2 RF5VOL     CHAR(6),   /* VOLUME SERIAL OF DEVICE        */
      2 *          CHAR(2),   /* RESERVED                       */
      2 RF5UNIT(2) CHAR(3),   /* UNIT ADDR FOR SD 1 AND 2       */
      2 RF5SLEN    FIXED(15), /* LENGTH OF THIS ENTRY           */
      2 RF5SCNT(2),           /* CACHING SUBSYSTEM COUNTS       */
        3 RF5SDID CHAR(1),    /* STORAGE DIRECTOR ID            */
        3 RF5DVID CHAR(1),    /* DEVICE ADDRESS                 */
        3 *       CHAR(2),    /* RESERVED                       */
        3 RF5SRCR FIXED(32),  /* SEARCH READ CACHING REQUESTS   */
        3 RF5SRCRH FIXED(32), /* SEARCH READ CACHING HITS       */
        3 RF5WCR  FIXED(32),  /* WRITE CACHING REQUESTS         */
        3 RF5WCRH FIXED(32),  /* WRITE CACHING REQUEST HITS     */
        3 RF5RSR  FIXED(32),  /* READ SEQUENTIAL REQUESTS       */
        3 RF5RSRH FIXED(32),  /* READ SEQUENTIAL REQUEST HITS   */
        3 RF5WSR  FIXED(32),  /* WRITE SEQUENTIAL REQUESTS      */
        3 RF5ICLR FIXED(32),  /* INHIBIT CACHED LOADING         */
                              /* REQUESTS                       */
        3 RF5RRBT FIXED(32),  /* READ REQ BOUND TO TRACK        */
        3 RF5WRBT FIXED(32),  /* WRITE REQUESTS BOUND TO TRACK  */
        3 RF5DCTO FIXED(32),  /* DASD TO CACHE TRANSFER OPS     */
        3 RF5CCDT FIXED(32),  /* CACHE TO CHANNEL DATA XFR      */
        3 RF5DCDT FIXED(32),  /* DASD TO CHANNEL DATA XFR       */
        3 RF5CDDT FIXED(32),  /* CACHE TO DASD DATA XFR         */
        3 RF5DCCDT FIXED(32), /* DASD TO CACHE DATA XFR         */
        3 *       CHAR(16),   /* RESERVED                       */
    1 SSTATUS BDY(WORD) BASED,/*                                */
      2 SSVOL     CHAR(6),    /* VOLUME SERIAL NUMBER           */
      2 *         CHAR(2),    /*                                */
      2 SSUNT1    CHAR(3),    /* FIRST UNIT ADDRESS             */
      2 SSUNT2    CHAR(3),    /* SECOND UNIT ADDRESS            */
      2 SSLN      FIXED(15),  /* LENGTH (40)                    */
      2 SSTAT,                /* STATUS DATA                    */
        3 SSCSD   CHAR(1),    /* SD ID                          */
        3 SSDEV   CHAR(1),    /* DEVICE ADDRESS                 */
        3 *       CHAR(2),    /* RESERVED                       */
        3 SSCS    BIT(8),     /* CACHING STATUS                 */
          4 SSHT  BIT(1),     /* EXPLICIT HOST TERMINATION      */
          4 SSIS  BIT(1),     /* INTERNAL ERROR TERMINATION     */
          4 *     BIT(6),     /* RESERVED                       */
        3 *       BIT(8),     /* RESERVED                       */
        3 SSDCS   BIT(32),    /* DEVICE CACHING STATUS BIT MAP  */
```

```
   3 SSCONF  BIT(32),    /* CONFIGURED SUBSYSTEM STORAGE
   3 SSAVAIL BIT(32),    /* AVAILABLE SUBSYSTEM STORAGE
   3 SSBOUND BIT(32),    /* BOUND SUBSYSTEM STORAGE
   3 SSOFFL  BIT(32),    /* OFFLINE SUBSYSTEM STORAGE
   3 *       CHAR(14);   /* RESERVED
```

Note: For more detailed description of the above fields, refer to the Sense Subsystem Counts and Status sections of

GA32-0067 IBM 3880 Storage Control Model 13 Description

```
/*****************************************************************/
/*                                                               */
/*  FORMAT OF SMF RECORD FOR CACHING SUBSYSTEM STATISTICS        */
/*                                                               */
/*****************************************************************/
 DCL 1 SMFF7 BASED(SMFF7PTR) BDY(BYTE),/*                        */
                             /* CACHING SUBSYSTEM RECORD        */
    2 SMFF7LEN    FIXED(16), /* LENGTH OF RECORD                */
    2 *           FIXED(16), /* SEGMENT DESCRIPTOR              */
    2 SMFF7SYS    BIT(8) ,   /* SYSTEM INDICATOR                */
    2 SMFF7TYP    FIXED(8),  /* RECORD TYPE                     */
    2 SMFF7TME    FIXED(32), /* TIME SINCE MIDNIGHT IN .01      */
                             /* SECOND UNITS                    */
    2 SMFF7DTE    CHAR(4),   /* DATE                            */
      3 SMFF7YR   FIXED(16), /* YEAR                            */
      3 SMFF7DAY  FIXED(16), /* DAY                             */
    2 SMFF7SID    CHAR(4),   /* SYSTEM ID                       */
    2 SMFF7LVL    FIXED(8),  /* CACHE RMF REPORTER LEVEL        */
    2 *           FIXED(8),  /* RESERVED                        */
    2 SMFF7NDV    FIXED(16), /* NUMBER OF DEVICE ENTRIES        */
    2 SMFF7MDL    FIXED(8),  /* CACHING SUBSYSTEM MODEL         */
    2 *           FIXED(8),  /* RESERVED                        */
    2 SMFF7NSS    FIXED(16), /* NUMBER SUBSYS STATUS ENTRIES    */
      3 SMFF7RTN  FIXED(16), /* IDCSS01 RETURN CODE IF NDV=0    */
    2 SMFF7INT    FIXED(32), /* NUMBER SECONDS SINCE SUBSYSTEM  */
                             /* STATISTICS LAST REPORTED        */
    1 RF7GDATA    BASED,     /* DATA PORTION                    */
/*****************************************************************/
/*                                                               */
/*  FORMAT OF DATA PROVIDED FOR CACHING SUBSYSTEM                */
/*                                                               */
/*****************************************************************/
      2 RF7VOL    CHAR(6),   /* VOLUME SERIAL OF DEVICE         */
      2 *         CHAR(2),   /* RESERVED                        */
      2 RF7UNIT(2) CHAR(3),  /* UNIT ADDR FOR SD 1 AND 2        */
      2 RF7SLEN   FIXED(15), /* LENGTH OF THIS ENTRY            */
      2 RF7SCNT(2),          /* CACHING SUBSYSTEM COUNTS        */
        3 RF7SDID CHAR(1),   /* STORAGE DIRECTOR ID             */
        3 RF7DVID CHAR(1),   /* DEVICE ADDRESS                  */
        3 *       CHAR(2),   /* RESERVED                        */
        3 RF7SRCR FIXED(32), /* SEARCH READ CACHING REQUESTS    */
        3 RF7SRCRH FIXED(32), /* SEARCH READ CACHING HITS       */
        3 RF7WCR  FIXED(32), /* WRITE CACHING REQUESTS          */
        3 *       CHAR(4),   /* RESERVED                        */
        3 RF7RSR  FIXED(32), /* READ SEQUENTIAL REQUESTS        */
        3 RF7RSRH FIXED(32), /* READ SEQUENTIAL REQUEST HITS    */
        3 RF7WSR  FIXED(32), /* WRITE SEQUENTIAL REQUESTS       */
        3 RF7ICLR FIXED(32), /* INHIBIT CACHE LOADING REQS      */
        3 *       CHAR(4),   /* RESERVED                        */
        3 *       CHAR(4),   /* RESERVED                        */
        3 RF7DCTO FIXED(32), /* NORM CACH DASD TO CACHE XFR OP  */
        3 RF7CCDT FIXED(32), /* CACHE TO CHANNEL DATA XFR       */
        3 RF7DCDT FIXED(32), /* DASD TO CHANNEL DATA XFR        */
        3 *       CHAR(4),   /* RESERVED                        */
        3 RF7DCCDT FIXED(32), /* DASD TO CACHE DATA XFR         */
        3 RF7DCCT  FIXED(32), /* DASD TO CHAN & CACHE DATA XFR  */
        3 RF7BPCR  FIXED(32), /* BYPASS CACHE REQUESTS          */
        3 RF7SCTO  FIXED(32), /* SEQ DASD TO CACHE XFR OPS      */
        3 *       CHAR(4),   /* RESERVED                        */
    1 GSTATUS     BDY(WORD) BASED,/*                            */
      2 GSVOL     CHAR(6),   /* VOLUME SERIAL NUMBER            */
      2 *         CHAR(2),   /* RESERVED                        */
      2 GSUNT1    CHAR(3),   /* FIRST UNIT ADDRESS              */
      2 GSUNT2    CHAR(3),   /* SECOND UNIT ADDRESS             */
      2 GSLN      FIXED(15), /* LENGTH (40 BYTES PER SD)        */
      2 GSTAT(2),            /* STATUS DATA                     */
        3 GSCSD   CHAR(1),   /* SD ID                           */
        3 GSDEV   CHAR(1),   /* DEVICE ADDRESS                  */
        3 *       CHAR(2),   /* RESERVED                        */
        3 GSCS    BIT(8),    /* CACHING STATUS                  */
          4 GSHT  BIT(1),    /* EXPLICIT HOST TERMINATION       */
          4 GSIS  BIT(1),    /* INTERNAL ERROR TERMINATION      */
```

```
    4 GSLI   BIT(1),     /* LIMITED CACHING STATE          *
    4 GSMS   BIT(1),     /* MODE SWITCH IS ACTIVE          *
    4 GSD0   BIT(1),     /* DASD CONTROLLER 0 AVAILABLE    *
    4 GSD1   BIT(1),     /* DASD CONTROLLER 1 AVAILABLE    *
    4 *      BIT(2),     /* RESERVED                       *
  3 *        BIT(8),     /* RESERVED                       *
  3 GSDCS    BIT(32),    /* DEVICE CACHING STATUS BIT MAP  *
  3 GSCONF   BIT(32),    /* CONFIGURED SUBSYSTEM STORAGE   *
  3 GSAVAIL  BIT(32),    /* AVAILABLE SUBSYSTEM STORAGE    *
  3 *        CHAR(4),    /* RESERVED                       *
  3 GSOFFL   BIT(32),    /* OFFLINE SUBSYSTEM STORAGE      *
  3 *        CHAR(14);   /* RESERVED                       *
```

Note: For more detailed description of the above fields, refer to the Sense Subsystem Counts and Status sections of

GA32-0083 IBM 3880 Storage Control Model 23 Description

BALANCED EXPONENTIAL APPROXIMATION ALGORITHM

Arun Kumar and Melur K. Raghuraman

Prime Computer Inc.,
500, Old Connecticut Path,
Framingham 01701

ABSTRACT

A new non-iterative approximate formula called The Exponential Approximation is presented to estimate statistical performance of single class, fixed rate, closed product form queueing networks. This approximation is combined with Balance Job Bounds to develop a new non-iterative algorithm called The Balance Exponential Approximate Algorithm. This algorithm is more accurate than both the Balance Job Bounds and the Exponential Approximation. This algorithm has the accuracy and the speed to be attractive for applications requiring little computation and fair accuracy.

1. Introduction

Closed Product Form Queuing Networks are widely used in models of computer systems. A number of approximate solution techniques exist for solving such networks [1, 2]. Most of the approximate solution techniques are based on iterative algorithms that are often expensive in terms of computation and storage requirements. For many practical applications, a very high degree of accuracy is not required. Often analysts require a "back-of-the-envelop" type of a technique to answer various bottleneck related questions. The iterative algorithms may be "over-powered" for such applications. Clearly, a simple and an easy to use technique for solving queueing networks is needed. This paper describes one such technique for single class, fixed rate queueing networks.

The Balance Job Bounds (BJB), as described by Zahorjan [2], meets some of the criteria outlined above. As the name implies, Balance Job Bounds give bounds on performance. These bounds can be used as estimates of performance. They are very accurate when the system is balanced, i.e., when the service demand at all centers is equal. Unfortunately, their accuracy is not acceptable when they are used to estimate performance of systems that are not balanced.

We describe an approximate technique called the Exponential Approximation, whose results are fairly accurate for unbalanced system. Then we show a method of combining the BJBs and the Exponential Approximation such that the combined solution is more accurate than the two parent techniques.

2. Balance Job Bounds (BJB)

The Balanced Job Bounds estimate the upper and the lower bound of throughput and response time for single class, fixed rate queueing networks.[1]

[1]In this paper, the term 'queueing network' refers to a single class, fixed rate queueing network.

Consider a network of k queues. Let D_l be the service requirement at center l, D_{max} be the largest service requirement among all centers, D_t be the sum of all D_ls and D_{avg} be the average service requirement. If there are n jobs in the queueing network then the Upper Balance Job Bound (UBJB) and the Lower Balance Job Bound (LBJB) for throughput are specified by equation (1) and (2) respectively.

$$UBJB(n) = Minimum(1/D_{max}, \ n \ / \ [D_t + (n-1)D_{avg}]) \quad (1)$$

$$LBJB(n) = n \ / \ [D_t + (n-1)D_{max}] \quad (2)$$

The UBJB is close to the exact solution for nearly balanced systems. The LBJB, on the other hand, is accurate when the load is equally concentrated at k1 (k1 <= k) centers with little service requirement at the remaining (k-k1) centers. The Balance Job Bounds have poor accuracy for unbalanced systems.

2.1. Examples

We illustrate BJB using the following two examples. We shall use these examples again later in the paper. The first example considers a queueing network that is nearly balanced. The second one considers a queueing network that is not balanced. Both systems have one CPU, three disks and eight batch jobs. It is required to estimate the throughput of these system.

Queueing Network A

Center	CPU	Disk1	Disk2	Disk3
ServiceDemand	0.24	0.27	0.23	0.26

Queueing Network B

Center	CPU	Disk1	Disk2	Disk3
ServiceDemand	0.4	0.1	0.2	0.3

Solution Queueing Network A

```
The throughput using the exact MVA technique [3]
was computed to be 2.893.

EXACT(8) = 2.893
```

Let us obtain the throughput estimated by the Balanced Job Bounds.

$D_{max} = 0.27$

$D_{min} = 0.23$

$D_t = 0.24 + 0.27 + 0.23 + 0.26 = 1.0$

$D_{avg} = 1.0/4 = 0.25$

$n = 8$

Substituting these values in equations 1 and 2, we have,

UBJB(8) = 2.909 error = 0.556%

LBJB(8) = 2.768 error = 4.315%

Solution Queueing Network B

The throughput using exact MVA technique [3] was computed to be 2.384

EXACT(8) = 2.384

Let us obtain the throughput estimated by the balanced job bounds.

$D_{max} = 0.4$

$D_{min} = 0.1$

$D_t = 0.40 + 0.10 + 0.20 + 0.30 = 1.00$

$D_{avg} = 1.0/4 = 0.25$

$n = 8$

Substituting these values in equations 1 and 2, we have,

UBJB(8) = 2.500 error = 4.889%

LBJB(8) = 2.105 error = -11.673%

These examples illustrate, as has been earlier stated, that balanced job bounds are accurate for balanced systems, but they may not be sufficiently accurate for unbalanced systems.

3. Exponential Approximation (EXAP)

The Exponential Approximation technique is based on two simple observations. The throughput of a queueing network is zero when the number of jobs in the network is zero. It asymptotically approaches $1 / D_{max}$ as the number of jobs in the network increase. We use an exponential equation (3) to fit this description of the throughput curve.

$$T(n) = EXAP(n) = a[1 - \exp(-b*n)] \quad (3)$$

where a and b are constants, and n denotes the number of jobs in the network.

The throughput when there is 1 job is $1 / D_t$ and the throughput when there are infinite jobs is $1 / D_{max}$. These values can be used to determine the constants a and b.

Solving for a and b and substituting the result back in equation (3) we obtain:

$$EXAP(n) = (1 / D_{max}) [1 - (1 - D_{max}/D_t)^n] \quad (4)$$

Equation (4) is the exponential approximation for the throughput. The response time R(n), when there are n jobs in the network, can be obtained using equation (4) and the Little's Law.

$$R(n) = n / T(n) \quad (5)$$

3.1. Examples

We obtain throughput for Queueing Networks A & B using the exponential approximation.

Queueing Network A :

EXAP(8) = 3.405 error = 17.698 %

Queueing Network B :

EXAP(8) = 2.458 error = 3.126 %

The results for queueing networks A and B are summarized in Table 1.

TABLE 1
Percent Error Table

	LBJB	UBJB	EXAP
Queueing Network A :	4.315%	0.556%	17.698%
Queueing Network B :	-11.673%	4.889%	3.126%

Table 1 illustrates an important relationship between the accuracy of the Balanced Job Bounds and the accuracy of the Exponential Approximation. We observe that when the Balance Job Bounds are accurate, the Exponential Approximation tends not to be accurate and vice-versa.

The following argument explains why the above property holds. It has been shown [2] that as the number of jobs in a queueing network approach infinity, the throughput of the network asymptotically approaches the inverse of the maximum service demand. That is, the center where the service demand is maximum has the greatest affect on the throughput of the queueing network.

The Balanced Job Bounds, by their very definition, are accurate when the service demand is same at all centers. However, as the network becomes unbalanced, the throughput of the queueing network tends to be largely influenced by the center with the maximum service demand. Both the Balanced Job Bounds reflect this fact and their estimates become accurate as the number of jobs in the queueing network approaches infinity (or is very large). However, their accuracy is often not very good when there are moderate number of jobs in the system.

The equation (4) for exponential approximation is based only on the maximum service demand and the total service demand. When the system is balanced, there is no single dominating center. Therefore for balanced systems, exponential approximation is not very accurate. For unbalanced systems, where there is a dominating center, the exponential approximation tends to be quite accurate. We have observed this fact empirically but are unable to derive mathematical bounds on the accuracy of exponential approximation.

Thus it seems that the two techniques complement one another. This indicates that better accuracy may be obtained if somehow the two techniques could be combined.

4. Balance Exponential Approximation Technique (BEAT)

In this section we describe a technique that combines the Balanced Job Bounds with the Exponential Approximation.

Consider the following estimates of throughput for a fixed rate queueing network:

Formula1(n) = EXAP(n)
Formula2(n) = UBJB(n)
Formula3(n) = LBJB(n)
Formula4(n) = (UBJB(n) + LBJB(n))/2

Our goal is to develop a test procedure for a given queueing network, such that the output of this test procedure indicates one of the above four formulas to be used to estimate the throughput of that queueing network.

We observe that the exact throughput of a queueing network can be expressed in a closed form for a small number of jobs. In particular, the exact throughput of a queueing network when there are only two jobs in the queueing network is specified below:

$$EXACT(2) = 2Dt / (D_t^2 + \sum D_k^2) \quad (6)$$

In order to obtain the throughput of a queueing network with **n** jobs, first the throughput for that network with only two jobs is computed using each of the approximate formulas and using equation (6). The formula that estimates throughput with least error when there are only two jobs in the network is used to estimate throughput for that network when there are **n** jobs in that network.

4.1. Examples

We compute the throughput of queueing network A and B using BEAT.

Queueing Network A

First compute the throughput using each of the four formulas and using equation (6) when there are only two jobs in the network.

Formula	Throughput	Error
EXACT(2)	1.599	0.000
Formula1(2)	1.730	0.131
Formula2(2)	1.600	0.001
Formula3(2)	1.575	-0.024
Formula4(2)	1.587	-0.011

Formula2 (that is UBJB) has least error when there are only two jobs in the network. Therefore the Upper Balance Job Bound is used to compute the throughput when there are eight jobs in the network. This has already been computed in section 2.1 to be 2.909. The exact throughput when there are eight jobs is 2.893. Thus the error using BEAT is 0.556%.

Queueing Network B

The throughput when there are only two jobs in the network is computed using the four formulas.

Formula	Throughput	Error
EXACT(2)	1.539	0.000
Formula1(2)	1.600	0.062
Formula2(2)	1.600	0.062
Formula3(2)	1.429	-0.110
Formula4(2)	1.514	0.024

Formula4 gives least error when there are only two jobs in the network. Therefore formula4 is used to estimate throughput when there are eight jobs in the network.

UBJB(8) = 2.500
LBJB(8) = 2.105
Formula4(8) = (2.500 + 2.105)/2 = 2.303

The exact throughput when there are eight jobs in the network is 2.383. Thus the throughput estimated using BEAT has an error of 3.3%.

5. Accuracy and Validation

BEAT has been tested on a large number of queueing networks. The results are summarized in Table 2 and in Figure 1. The following test methodology was used. The number of jobs in the queueing network were varied from 5 to 20. For each job value, one

hundred random networks were generated. The number of centers in the network were randomly chosen from 2 to 50. For each random network generated, one hundred random samples of loading were generated. The service demand at each center varied randomly from 0.0 to 1.0 .

Figure 1 shows that BEAT is significantly more accurate than the Balanced Job Bounds and the Exponential Approximation. Although the maximum error for BEAT was observed to be 15.6 %, normally the results are more accurate. About 90% of the times, the error in throughput was observed to be less than 5%.

Table 3 indicates the frequency of selection of the four formulas in the tests conducted. It is interesting to note that about seventy percent of the times the average of the Balance Job Bounds is used. The Lower Balance Job Bound is very infrequently selected.

An additional point of interest in Figure 1 is that the Upper Balance Job Bound seems to be more accurate than the Lower Balance Job Bound. This appears to contradict Zahorjan's [4] observation that, in general, Lower Balance Job Bound is closer to the exact solution than the Upper Balance Job Bound. In our tests, we observed that the average error for Upper Balance Job Bound is less than the average error for Lower Balance Job Bound, but the maximum error for Upper Balance Job Bound is higher than the maximum error for Lower Balance Job Bound.

6. Conclusions

In this paper, a new technique to solve single class fixed rate queueing networks was presented. The accuracy of this technique is sufficient for many practical applications. As this is a non-iterative technique, the computation and memory requirements are minimal. Therefore this technique promises to be attractive for applications requiring little computation, fast response, and fair accuracy.

7. References

1. K.M. Chandy and D. Neuse, "Fast Accurate Heuristic Algorithm for Queueing Network Models of Computing Systems", CACM 25, 2 Feb 1982.
2. J. Zahorjan, K.C. Sevcik, D.L. Eager, and B.I. Galler, "Balance Job Bound Analysis of Queueing Networks", CACM 25, 2 Feb 1982.
3. M. Reiser and S.S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," J.ACM 27,2 pp. 313-322, April 1980.

Table 2 : Error Statistics

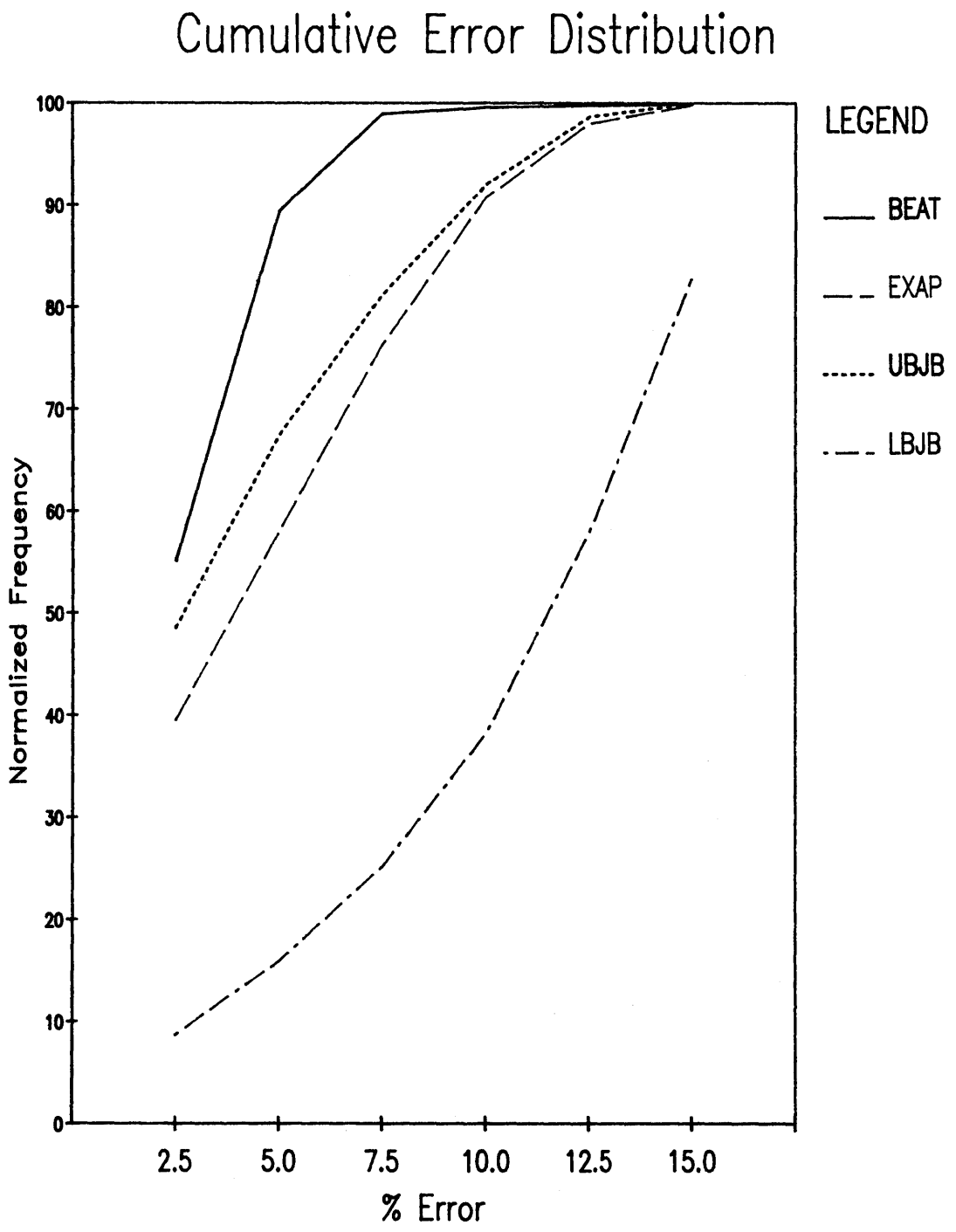
Number of tests = 160000

Method	% Mean Error	% Variance Error	% Maximum Error
BEAT	2.31%	3.8%	15.6%
EXAP	5.52%	21.3%	21.9%
LBJB	9.57%	20.6%	23.5%
UBJB	8.25%	53.1%	37.9%
(UBJB + LBJB)/2	2.52%	3.6%	17.5%

Table 3 : Frequency of Formula Selection

Formula	Frequency
Formula1 (EXAP)	13.36%
Formula2 (UBJB)	14.26%
Formula3 (LBJB)	0.26%
Formula4 (UBJB + LBJB)/2	71.72%

FIGURE 1


Cumulative Error Distribution
LEGEND
BEAT
EXAP
UBJB
LBJB
Normalized Frequency
100
90
80
70
60
50
40
30
20
10
0
2.5
5.0
7.5
10.0
12.5
15.0
% Error

SENSITIVITY ANALYSIS OF SYSTEM CHANGES UNDER A PRODUCTION ENVIRONMENT

T. L. LO
McDonnell Douglas
Aerospace Information Services Company

ABSTRACT

System changes such as migration and modification always require full functional compatibility between the old and new versions of the system. The performance impact of system changes, however, does not seem to receive the same attention as functional compatibility due mainly to its complex nature, especially under a production system environment in which workload varies. This paper discusses how to use a regression modeling approach to determine the level of acceptance of a system's changes under a production environment in which workload condition is not controlled. Several examples on MVS/XA, VTAM, and ISPF migrations are presented.

INTRODUCTION

When a change is introduced to a system software, hardware, application program, or workload profile, system performance is likely to be affected. A change can be, for example, a new release of an access method, such as IBM's Virtual Telecommunication Access Method (VTAM) or an operating system migration, such as IBM's Multiple Virtual Storage (MVS) system. Most DP installations evaluate the changes only from functional compatibility viewpoint in a standalone environment. As long as the same (or better) functions are provided, the evaluation is likely complete. When the changed system is put into production, there is a chance that performance can be noticeably altered by the changes which are tested for their functional compatibilities, but not for their performance impacts.

There are, however, installations that do both functional and performance testings on system changes. The most common approach is benchmarking, in which a set of jobs, representing a range of typical workloads, are executed with and without the changes in a standalone testing environment. Thus, the workload is controllable. Although the evaluation of changes is straightforward in that the difference in performance between the new and old versions can simply be reported, the success of this approach depends on how well the benchmark represents the real workload, and how one interprets the results. Examples of this type of approach are synthetic benchmarks and the teleprocessing network simulator (TPNS).

The synthetic benchmark approach (BUCH69) utilizes one program with adjustable input parameters, such as number of CPU loops and number of I/Os, etc. running as a batch job in an attempt to represent the characteristics of certain production workloads. It requires fewer resources and can be executed many times within a fairly short period of time. The TPNS approach is designed to simulate the performance of on-line applications, such as IMS and TSO, using a set of properly designed scripts to represent the unique characteristics of the application it simulates. It requires physical facilities such as computer systems, front-end processors, and disk storage, and also requires manual manipulation.

The other approach, which draws much less attention than benchmarking, is to evaluate the changes on the performance of a real workload under a production system. That is, the system is run with and without the changes under an uncontrolled workload environment after the functions of the changed system are verified. Evaluation of performance impact in an uncontrolled workload environment can be complicated since any change in the system may also be affected by a workload variation as well as the change itself. Therefore, it is necessary to determine to what extent the system performance impact is acceptable. Consequently, a number of performance indicators need to be identified in order to determine the degree of tolerance on performance impact.

Due to its complex nature, a statistical analysis approach seems to be an ideal tool for evaluating system changes. One technique is to use regression models which correlate variables describing performance indicators to variables describing the uncontrolled workload variation and system changes. This paper explores the applicability of regression models in determining the degree of tolerance on the performance impact possibly caused by system changes. The first section reviews the general regression models; the second section illustrates the application of these models on various production system and application changes with examples; the last section briefly discusses the pros and cons of this technique.

REGRESSION MODELS FOR SYSTEM CHANGE EVALUATION

A regression model used for evaluating system changes correlates two types of variables: workload variables that describe system changes and workload variations, and performance variables that describe performance indicators. Each of the workload and performance variables in a regression model can be specified at either the micro (detailed) or macro (composite) level. Since the micro level involves immense data collection, reduction, and analysis efforts, most regression models deal with only the composite level.

The general form of a regression model can be expressed as follows.

$$Y(j) = \sum_{i=0}^{N} A(i)X(i) \quad \text{for } j = 1,\ldots M$$

where X(i) is a workload variable that describes various workload and system conditions, A(i) is the unknown regression constant, Y(j) is the performance variable corresponding to the jth set of workload variables, N represents the number of workload variables, and M is the number of regression sets. The X(i)'s can be further grouped into two categories: one describes workload and the other describes the system environment. There are two basic steps when applying regression models: the first one considers system changes only and the second one considers both system changes and workload variations. Friedman and Waldbaum (FRIE75) pro-

vided a detailed description of regression The following discussions follow their general notations.

ANALYZE CHANGES WITHOUT WORKLOAD VARIATION

The first step in using a regression model is to temporarily ignore the workload variation and analyze the performance variables under various system conditions. For example, if the response time is chosen as a performance variable, then its cumulative distribution function under various system conditions can be examined using the averaged response time for the 50th, 90th, and 95th percentiles. If there is an observed difference between two system conditions, a least squares fitness model should be used to further explain the source of difference. The least squares estimate of the unknown regression constants, A(i), can be found by solving the normal equations expressed in a single matrix form:

$$X'XA = X'Y \quad \text{where } A = \begin{bmatrix} A(0) \\ \vdots \\ A(N) \end{bmatrix}$$

and X is a (M x N) matrix whose columns correspond to various system conditions, and X' is the transpose of X. A hypothetical example is used to illustrate the fitness model.

If there are three different system processing conditions represented by X(0), X(1) and X(2) variables with 35, 9 and 8 days of data collected, respectively, then

$$A = \begin{bmatrix} A(0) \\ A(1) \\ A(2) \end{bmatrix}$$

and X is a 35x3 matrix whose columns are X(0), X(1), and X(2) respectively. The fitness model matrix is then

$$\begin{bmatrix} 35 & 18 & 8 \\ 18 & 18 & 0 \\ 8 & 0 & 8 \end{bmatrix} \begin{bmatrix} A(0) \\ A(1) \\ A(2) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{35} Y(i) \\ \sum_{i=1}^{18} Y(i) \\ \sum_{i=28}^{35} Y(i) \end{bmatrix}$$

Therefore, the following combinations of A(i)s indicate different system environments:

* A(0) + A(1) -- the first system condition
* A(0) -- the second system condition
* A(0) + A(2) -- the third system condition
* A(1) -- the difference between systems 1 and 2
* A(1) - A(2) -- the difference between systems 1 and 3
* A(2) -- the difference between systems 2 and 3

In order to test the statistical significance of the observed differences without considering workload variations, we need to make further assumptions that the unexplained variations are distributed uniformly, with zero mean and variance equal to a constant that can be estimated from the residue, which is the difference between observed and fitted values. Therefore, the student-t test can be used on the basis of the estimated standard error of a coefficient. A student-t is equal to the ratio of the coefficient to its standard error. If there is strong evidence indicating that a real difference exists between two system conditions, the result can substantiate the observation that two system conditions are statistically different.

ANALYSIS OF WORKLOAD WITH WORKLOAD VARIATIONS

The second step is to include workload variations in data analysis in order to determine whether the observed performance disparity can be accounted for by the workload variation. Furthermore, if there is a difference, can it be explained by both types of variables ? Thus, the regression model is required to adequately correlate both workload and system conditions to performance indicators.

The complexity of this approach lies in determining whether the observed differences of performance indicators are constant under any two system conditions and any workload. If so, we may be able to assume that the effect of the system condition is independent of the workload variation. This can be studied using parallel regression models.

On the other hand, if the parallelism does not exist; that is, the difference is not a constant under any two system conditions and any workload, then non-parallel regression models are required. Since non-parallel regression involves more complexity (i.e. more sophisticated regression equations and verifications on model adequacy, etc.), this paper only explores parallel regression models. While it should be noted, however, that even parallel regression model may not be entirely appropriate in certain cases, they still can provide some explanations on performance differences.

The general parallel regression model has the following form:

$$Y = A(0) + \sum_{i=1}^{N} A(i)X(i)$$

Based on the way the system conditions are arranged, the general equation can be arranged into a set of models, each corresponding to a specific system condition. Using previous hypothetical example, the general model can be represented by the following models:

System cond. 1: $Y(1) = A(0) + A(1) + A(3)X(3) + \ldots$.
System cond. 2: $Y(2) = A(0) + A(3)X(3) + \ldots$.
System cond. 3: $Y(3) = A(0) + A(2) + A(3)X(3) + \ldots$

Again, A(0) represents the effect of system condition #2, A(1) represents the difference between system conditions #1 and #2, and A(2) represents the difference between system conditions #2 and #3. Note that other coefficients (e.g. X(4),...X(N)) correspond exclusively to workload variables, and are the same for different conditions, based on the parallel regression assumption. Therefore, this set of models enables us to study the effect of performance difference due to system conditions separate from workload variation.

Assuming this model is correct, then several common statistical fitting techniques can be used. The student-t test can examine each coefficient for different system conditions. Note that the effect of workload variables is now considered, and if the effect can be accounted for by workload alone, the t-statistic corresponding to a specific coefficient would be insignificant.

The other fitness parameter that is worth considering is R-square, the multiple correlation coefficient which describes the proportion of variation accounted for by the fitted regression model. If the parallelism assumption is correct, the R-square value would indicate

the adequacy of the model if there is a real difference between any two conditions.

Another useful statistic for checking model adequacy is the residual sum of squares which attempts to provide an unbiased estimate of the unknown variance of the fitness errors. By plotting the residuals, the outlying observations, systematic departure from randomness, non-constancy of variance, and nonnormality can all be examined.

MODEL APPLICATION AND CASE STUDIES

CASE STUDY #1 - MVS/XA MIGRATION

When IBM's MVS/XA operating system was announced, several of its advertised performance improvements had already convinced many DP installations to take advantage of MVS/XA without knowing what the real impact it would have. The migration study here (KELL83) involved an IBM 3083 model J processor with 24 I/O channel and 24 MB of real memory. The operating system was MVS/SP to be replaced by MVS/XA.

Three sets of benchmark jobs were chosen to measure the performance difference. The first set consisted of batch jobs representing a wide range of production application programs such as Assembly, catalog, utility, SORT, DASD, VSAM and various language compilers. The second set consisted of a group of selected on-line application programs such as SAS, SPF, MARK IV, etc. The third set was the real time application such as IMS and CICS.

The system conditions were naturally MVS/SP and MVS/XA, represented by X(0) and X(1) respectively. System execution time (Y) was selected as The performance variable with 50th (Y(1)) and 95th (Y(2)) percentiles. The workload variables were CPU time (X(3)), I/O time (X(4)), and multiprogramming level (X(5)). For the real-time application, the number of transactions per hour, X(6), was added to the workload variables.

The prime shift hours (0800-1700) of weekdays were used to collect data on performance and workload variables with respect to different system conditions. Data of two weeks before and one week after the MVS/XA migration were collected. Although it did not seem sufficient to satisfy the statistical significance, that was all the time given without impacting the production environment. Data collected can be grouped into a matrix form such as the following based on workload and performance variables and system conditions:

X(1)	X(2)	X(3)	X(4)	X(5)	X(6)	Y(1)	Y(2)
0	1						
0	1						
0	1						
1	0						
1	0						
1	0						

where X(1) = 0 and X(2) = 1 represent the MVS/SP condition and X(1) = 1 and X(2) = 0 represent the MVS/XA condition.

The first step of changed system evaluation is to temporarily ignore workload variables. The following tables summarize the analyses. Table 1 shows the batch benchmark results. Y(1) and Y(2) represent 50th and 95th percentiles of the number of runs of an application respectively.

X(1)	X(2)	Y(1) HH.MM.SS	Y(2) HH.MM.SS	Application name	# of runs
0	1	00.02.83	00.09.06	ASM	20
1	0	00.02.21	00.08.53		
0	1	00.01.38	00.04.50	catalog	16
1	0	00.01.40	00.04.62		
0	1	00.03.30	00.07.02	COBOL	10
1	0	00.03.27	00.06.87		
0	1	00.03.60	00.06.95	FORTRAN	6
1	0	00.03.61	00.06.98		
0	1	00.17.58	00.38.61	Utility	104
1	0	00.17.23	00.37.88		
0	1	00.52.07	01.17.93	SORT	18
1	0	00.52.07	01.18.79		
0	1	00.02.00	00.03.38	CHK/RST	4
1	0	00.02.00	00.03.38		
0	1	00.18.98	00.34.67	DASD	94
1	0	00.19.83	00.35.87		
0	1	00.14.71	00.30.67	SMP	22
1	0	00.12.83	00.32.91		
0	1	00.20.17	00.48.71	OEM	74
1	0	00.19.26	00.32.91		
0	1	00.00.53	00.01.38	VSAM	12
1	0	00.00.51	00.01.28		
0	1	00.09.06	00.20.41	PL/I	42
1	0	00.09.41	00.21.63		
0	1	00.01.89	00.02.96	ISAM	10
1	0	00.02.01	00.03.01		

Table 1 Batch Benchmark Results

Table 2 shows the summary of the selected interactive benchmark comparison and Table 3 shows the real-time IMS and CICS application timing comparison.

X(1)	X(2)	Y(1) HH.MM.SS	Y(2) HH.MM.SS	Application name	# of runs
0	1	00.05.11	00.09.36	PL/I CLG	55
1	0	00.05.12	00.09.14		
0	1	00.04.68	00.07.90	ASM LG	43
1	0	00.05.08	00.07.92		
0	1	00.02.05	00.03.11	MARK IV	18
1	0	00.02.03	00.03.08		
0	1	00.04.03	00.07.54	SAS	130
1	0	00.04.02	00.07.26		
0	1	00.01.05	00.01.81	SPF	991
1	0	00.01.08	00.01.78		
0	1	00.06.64	00.12.06	RAMIS II	17
1	0	00.06.06	00.11.41		
0	1	00.08.70	00.11.12	FORT CLG	13
1	0	00.08.73	00.11.41		
0	1	00.02.10	00.05.01	COBOL CLG	11
1	0	00.02.07	00.04.27		

Table 2 Interactive Benchmark Results

X(1)	X(2)	Y(1) MM.SS.MS	Y(2) MM.SS.MS	Application name
0	1	00.02.10	00.04.61	IMS
1	0	00.02.05	00.04.60	
0	1	00.03.06	00.04.07	CICS
1	0	00.02.92	00.04.08	

Table 3 Real Time Benchmark Results

As highlighted in Table 1, there were observable differences in execution time of DASD, SMP, OEM and possibly PL/I application programs. It is necessary to test

their statistical significance in order to determine whether these differences are caused by workload variation. The general least squares fitted model is

$$Y = A(0)X(0) + A(1)X(1)$$

or in a single matrix form,

$$\begin{bmatrix} 15 & 5 \\ 5 & 0 \end{bmatrix} \begin{bmatrix} A(0) \\ A(1) \end{bmatrix} = \begin{bmatrix} \sum_{1}^{15} Y(i) \\ \sum_{10}^{15} Y(i) \end{bmatrix}$$

Therefore, A(0) represents MVS/XA condition and A(0) - A(1) represents MVS/SP condition. In order to test the statistical significance, we need to further assume that the unexplained differences are normally distributed, with zero mean and variance equal to a constant that can be estimated from residuals.

The t-statistics corresponding to the coefficient A(1) for the 50th and 95th percentiles are:

	DASD	SMP	OEM	PL/I
50%	-0.73	1.19	1.71	-0.09
95%	-0.91	1.77	3.04	-0.92

Since t > 2, there is strong evidence that there is a real difference between MVS/SP and MVS/XA on the OEM application whereas SMP programs only showed a marginal difference.

Although the t-test indicated that there was a real difference on the OEM application, we need to expand the analysis in order to include workload variables to determine whether the effect we observed could be attributed to workload variation. Thus, the parallel regression model is needed.

Recall that the parallel regression model assumes that, if the model is correct, the difference in performance under any two system conditions and any workload is independent of the workload. Since A(1) represents the difference between MVS/XA and MVS/SP, the t-statistic associated with A(1) can be used to examine the model's assumption. That is, if the difference can be accounted for by workload alone, the t-statistic corresponding to A(1) would be insignificant.

The parallel regression model consists of two equations corresponding to 50th and 95th percentiles of the execution time.

$$Y(1) = A(0) + A(3)X(3) + A(4)X(4) + A(5)X(5)$$
$$Y(2) = A(0) + A(1) + A(3)X(3) + A(4)X(4) + A(5)X(5)$$

where X(3) is OEM CPU time, X(4) is the number of I/Os, and X(5) is the multiprogramming level.

The original data of 74 OEM runs were used to yield t-statistics and R-square value for the 50th and 95th percentile response times between MVS/XA and MVS/SP conditions. Table 4 shows the summary of the coefficients, t-statistics, and R-square of the fitted model.

	Y(1)	Y(2)
A(0)	0.218	0.233
A(1)	0.449	0.988
t(A(1))	3.171	5.617
R-square	0.61	0.57

Table 4 OEM Statistics with Workload Variables

Since the t-statistic corresponding to A(1) indicates that the difference is significant and the R-square value does not show a strong correlation of variation accounted for by the fitted model, the workload variables alone are not responsible for the difference. Further investigation revealed that there were two incompatible local modifications which caused an adverse effect (longer execution time) under MVS/XA.

CASE STUDY #2 - VTAM MIGRATION

ACF/VTAM release 3 with Extended Network Addressing (ENA) capability was designed to provide relief for those subarea craving installations and we were one of them. As usual, all functions under release 3 were tested for their compatibility but none of the performance impact was assessed. After the implementation of ACF/VTAM release 3 on one of the TSO systems (an IBM 3084Q with 96 MB of real memory under MVS/XA operating system), slow sporadic TSO response times were reported.

A TSO script was developed in an attempt to determine whether the new release was responsible for the slow response time problem. The script was executed from a locally attached terminal, a remote terminal, and a local terminal on a system running under VTAM release 2. The script was run twice each day for five days at each terminal at 1000 and 1400 hours which are the peak resource demanding periods in our environment.

Therefore, three system conditions were to be studied. A local terminal under VTAM release 3, a remote terminal under release 3, and a local terminal under VTAM release 2. A VTAM I/O trace was activated during script execution and RMF monitor was also used to trace the activity of that particular terminal in an attempt to correlate response time reported by RMF and VTAM.

The performance variables were average response time, network time, inbound VTAM time, outbound VTAM time, host processing time, and 90th percentile on total response time. The workload variables were number of transactions as seen by RMF and VTAM, CPU utilization, number of TSO address spaces, and average TSO user working set size. It should be pointed out that the network time here included data transmission time, user think time and/or typing time. Hence, it is not a truly data transmission time. The data matrix corresponding to performance and workload variables are shown in Table 5.

	X(1) X(2) 1 0	X(1) X(2) 0 0	X(1) X(2) 0 1
X(3) RMF # TX	59	59	59
X(4) VTAM #TX	58	58	58
X(5) System CPU%	88.1	79.3	89.2
X(6) TSO CPU%	43.3	9.7	44.7
X(7) #TSO ASID	183	13	190
X(8) TSO WSS (KB)	441	313	438
Y(1) IN VTAM (ms)	1.021	1.19	1.196
Y(2) OUT VTAM (ms)	0.923	0.49	0.931
Y(3) HOST time (sec)	0.912	0.851	1.195
Y(4) RMF resp time	0.927	0.807	0.975
Y(5) VTAM resp time	0.932	0.857	1.206
Y(6) RMF 90% R.T.	2.466	1.660	2.375
Y(7) VTAM 90% R.T.	2.59	1.751	2.453
Y(8) Network time	6.65	7.55	7.19

Table 5 TSO Script Timing & Resource Usage Statistics

Where X(1) = 1 and X(2) = 0 represent the system condition that local terminal user executing script under VTAM release 3; X(1) = 0 and X(2) = 1 represent the system condition that remote user executing script under VTAM release 3; X(1) = 0 and X(2) = 0 represent the system condition that local terminal executing the same script under a system running VTAM release 2.

As indicated in Table 5, the outbound VTAM time between The two releases was obviously different. Both average and 90th percentile response times reported by RMF and VTAM on the two releases were also different. A t-test statistic confirmed that there was a significant difference on outbound VTAM time between the two releases ($t > 4$), but no significant difference on local and remote terminals running under release 3 ($t < 0.5$). Further investigation revealed that system condition #2 (X(1) = 0 and X(2) = 0) was not a TSO system and the delays due to paging, swapping and device queuing were much smaller than the other two system conditions. Therefore, the response time difference between the two releases was attributed to workload variation.

The outbound VTAM time was investigated further since the difference was significant. It was discovered that the default value for the pacing parameter was different in the two releases. When the load is less intensive, the queuing time between host and outbound VTAM interface is insignificant. As the load increases, so does queuing time, although not proportionally.

CASE STUDY #3 - TSO & ISPF MIGRATION

The Interactive System Productivity Facility (ISPF) operates under Time Sharing Option (TSO) to allow users to develop, test, and execute programs interactively. It is a manually driven facility to assist users with many programming tasks in developing applications. The TSO system is an IBM 3084Q with 96 MB of real memory under THE MVS/XA operating system. The average number of logged on TSO users was 270 including 20 CAD/CAM users. The other workload included a small test CICS system and a small IMS test system. Batch jobs ran as a background application.

In order to utilize the virtual storage constraint relief offered by MVS/XA, and additional functions under the new release of ISPF, a decision was made to replace TSO/E 1.0 with TSO/E 2.1 and ISPF 1.1.0 with ISPF 2.1.2. A set of TSO scripts were developed to study the performance impact.

Script #	Description
1A	TSO/E basic functions (no ISPF)
2A	ISPF/PDF script for ISPF 1.1.0 no TSO/MON ISPF sysevents
2B	ISPF/PDF script for ISPF 2.1.2 no TSO/MON ISPF sysevents
2C	ISPF/PDF script for ISPF 2.1.2 include TSO/MON sysevents concatenated ISPF/PDF libraries
2D	ISPF/PDF script for ISPF 2.1.2 combined ISPF/PDF libraries No TSO/MON sysevents
2E	ISPF/PDF script for ISPF 2.1.2 like 2D with added ISPCTL0 in log-on procedure; recovery turned off.

The performance variables were:

* TSO trivial transaction response time
* TSO all transaction response time

The workload variables were:

* CPU %
* Paging (demand, swap and total)
* TSO transaction rate
* TSO TCB CPU time per transaction
* TSO total memory
* TSO average memory per resident user
* CAD/CAM average memory per resident user

There were five different changes associated with the study:

* (001) - TSO/E 1.0 & ISPF 1.1.0
* (010) - TSO/E 2.1 & ISPF 2.1.2
* (011) - TSO/E 2.1 & ISPF 2.1.2 with adding two swap data sets, removing one paging data set, adjusting SRM parameters, removing TSOMON ISPF interface
* (100) - Similar to #3 with modified log-on procedure to pre-allocate working data sets
* (101) - Similar to #4 with two PTF modifications, and increased system think time.

The detailed comparisons can be seen in (OBER86). The following tables only present the summarized results. Table 6 shows the summary of comparisons on five system conditions with respect to performance and workload variables.

System condition	X(1) =	0	0	0	1	1
	X(2) =	0	1	1	0	0
	X(3) =	1	0	1	0	1
CPU util. %		84.4	89.6	84.6	83.7	84.2
Demand paging		52.0	107.0	43.0	38.0	40.0
Swap paging		63.0	111.0	243.0	233.0	172.0
Total paging		130.0	231.0	296.0	280.0	220.0
TSO TX rate		13.9	14.0	13.1	12.9	13.0
Triv. TX resp.		0.39	0.43	0.40	0.36	0.36
Total TX resp.		0.93	1.26	1.17	1.03	1.02
TSO TCB CPU/TX(ms)		57.0	73.0	71.0	75.0	73.0
TSO EXCP/TX		12.36	13.61	14.20	13.96	13.01
total memory(MB)		37.32	42.33	31.36	30.05	33.97
TSO avg WSS (KB)		328.0	360.0	453.0	440.0	441.0
CAD/CAM WSS (KB)		824.0	754.0	925.0	934.0	937.0

Table 6 TSO/E & ISPF Migration Data Comparison

Since the paging activities, TCB CPU/TX, and average WSS were significantly different between the two TSO/E and ISPF releases, further analyses were conducted to determine if the difference could be explained. Table 7 shows further CPU, memory and I/O comparison on the two different releases with respect to various scripts described above.

	TCB+SRB Sec.		TCB Sec.		EXCPs		Memory	
TSOE	1.0	2.1	1.0	2.1	1.0	2.1	1.0	2.1
1A (TSOE)	1.89	1.98	1.77	1.88	269	270	196	220
2A/2B (ISPF)	2.46	3.75	2.21	3.47	355	476	251	315
2A/2C (ISPF)	2.46	3.74	2.21	3.46	355	476	251	327
2A/2D (ISPF)	2.46	3.62	2.21	3.35	355	446	251	326
2A/2E (ISPF)	2.46	3.45	2.21	3.18	355	427	251	326

Table 7 CPU, Memory & EXCP Comparison on TSO/E & ISPF

Table 8 shows the PDF function comparison between 2A and 2B (ISPF 1.1.0 vs 2.1.2) and 2A and 2E (ISPF 1.1.0 vs 2.1.2 with a modified logon PROC, no TSO/MON and no recovery).

Function	TCB+SRB Sec. 1.0	2.1	2.1	TCB Seconds 1.0	2.1	2.1	EXCPs 1.0	2.1	2.1
Browse	0.184	0.277	0.285	0.144	0.237	0.235	33	32	31
Edit	0.530	0.842	0.720	0.462	0.763	0.650	94	128	114
3.1	0.128	0.178	0.175	0.108	0.160	0.160	19	21	21
3.2	0.344	0.432	0.430	0.312	0.398	0.395	40	45	45
3.8	0.584	1.22	1.22	0.520	1.140	1.140	54	119	116

Table 8 PDF Function Comparisons for TSO/E and ISPF

Since the scripts were executed only once in each condition, the normal t-statistic could not be applied, nor any other regression models. Therefore, the only way was to manually observe the difference in response time.

The response times between TSO/E 1.0 & ISPF 1.1.0 and TSO/E 2.1 & ISPF 2.1.2 releases were different (35% in total and 10% in trivial transactions). As tuning changes were introduced to the new release, the difference tends to become insignificant (< 10% in both types of transactions). However, the differences in workload variables were quite obvious (e.g. demand paging was doubled).

According to the vendor, the difference between the two TSO/E and ISPF releases was a trade-off consideration. The performance expectations from the new release of TSO/E and ISPF were:

* CPU time reduction
  - CLIST processing
  - Removal of local modifications
  - Library allocation
* CPU time increase
  - More libraries
  - TSO/MON interface
  - Authorized facility
  - ISPF 3.8 CLIST code execution
* Memory & I/O
  - Virtual storage constraint relief
  - Smaller working set size
  - more real memory for link pack area (LPA)
  - More I/O from ISPF 3.4
  - Less I/O to SYS1.BRODCAST
  - Smaller below-the-line private requirement for ISPF users

We observed following effects due to the new release of TSO/E and ISPF:

* Virtual storage - 584KB virtual storage constraint relief, 1364 KB increase in extended link pack area, and the net link pack area increase was 780 KB.
* Real storage - Link pack area increase of 464 KB, private area decrease of 2833 KB.
* Other areas - 28% increase in TSO TCB CPU per transaction; 10% increase in EXCPs per transaction; 76% increase in swap paging; 10% increase in average user working set size; and poor batch turnaround.

Again, the trade-off was to gain more functionality improvements and user friendly conversions while still being able to meet the service goal requirements.

CONCLUDING REMARKS

We have shown that traditional regression models can be used to study performance impacts due to systems changes under a production system environment where workload varies. We feel that this approach can provide useful information for an analyst to determine what is an acceptable level of tolerance between two or more different system conditions and what to do next after identifying the area of discrepancy.

The use of a regression model to study the performance impact due to system changes is a straightforward process. However, to interpret the results requires an understanding of basic statistical principles. In some case, the application of regression models is not as straightforward as it seems since there may not have sufficient data points to satisfy the validity of statistical analysis. Furthermore, the parallelism assumption may not be valid in certain cases even though it can provide useful information. When the real difference is not obviously observable, an automated tool would be desirable to eliminate some guess work errors.

REFERENCES

BUCH69 Buchholty, W., "A Synthetic Job for Measuring System Performance," IBM System Journal, Vol. 8, No. 4, 1969.

FRED81 Freund, R.J. & Littell, R.C., "SAS for Linear Models," SAS Institute, Cary, N.C., 1981

FRIE75 Friedman H. P. & Waldbaum, G., "Evaluating System Changes Under Uncontrollable Workloads: A Case Study," IBM System Journal, Vol. 14, No. 4, 1975

KELL83 Kelly, D. B., "An Installation and Migration Methodology for Operating System," Master's thesis, Dept. of Computer Science, University of Missouri at Rolla, May, 1983

OBER86 Obermeier, M. J., "Performance Impact Study on Different TSO/E and ISPF Releases Under MVS/XA," 1986 Spring St. Louis CMG Conference, St. Louis, April, 1986

SAS85 SAS User's Guide: Statistics, Version 5, SAS Institute, Cary, N.C., 1985

Characterizing UNIX® Stream Mechanism Via Pipe Experiments

T. Paul Lee
AT&T Bell Laboratories
Crawfords Corner Road
Holmdel, NJ 07733

ABSTRACT

A stream is a full-duplex connection between a user's process and a device or pseudo-device in the UNIX® system. It consists of several linearly connected processing modules, and is analogous to a UNIX shell pipeline, except that data flows in both directions. It is anticipated that new drivers and protocols will be implemented using the stream mechanism. This paper presents a simple characterization of this mechanism by experiments using stream-based pipes (SPIPEs). Factors varied in the course of the experiments were read/write system call message sizes, pass-through procedure call versus service scheduling, null service versus CRC-16 generation/verification service, and the number of modules pushed on the stream device.

We have found that, stream-based pipe has slightly higher throughput relative to a similarly configured pipe with message sizes up to 256 bytes, the default internal message size. Throughput beyond this point is hampered by the need to break up large messages. By setting a stream option to overwrite the default internal fragmentation for large messages, stream-based pipe throughput is also higher than that of regular pipe for very large message sizes.

To determine stream performance in a more realistic environment, a simple CRC-16 stream protocol was implemented. When simple table-driven CRC-16 generation/verification processing is performed on a per-message basis for each stream module, the throughput of the stream pipe is determined by CRC-16 calculation overhead for larger message sizes. This implies that with realistic message processing, the throughput of any stream device will be predominantly determined by the message processing overhead; the overhead induced by the basic stream mechanism is negligible.

1. *INTRODUCTION*

A stream is a full-duplex connection between a user's process and a device or pseudo-device in the UNIX®* system [1]. It consists of several linearly connected processing modules, and is analogous to a UNIX shell pipeline, except that data flows in both directions. Data flows are regulated by high and low watermarks associated with each module. Each stream processing module consists of a pair of queues, one for each direction. A queue comprises not only a data queue proper, but also two routines and some status information. One routine is the *put procedure*, which is called by its neighbor to place messages on its data queue. The other, the *service procedure*, is scheduled to execute whenever there is work for it to do. The objects passed between queues are pointers to blocks obtained from a memory allocator. Although each queue module behaves like a separate process, it is not a real process to the UNIX operating system. Queue processing routines do not block when they cannot proceed, but must explicitly return control.

This stream mechanism is an an important new feature of UNIX System V Release 3.0 [2,3]. New drivers and protocols will be using the stream mechanism in their implementations. It is important to have some quantitative insight and understanding on the usage of the stream mechanism. It is difficult, however, to characterize the stream mechanism by itself without the context of a driver or protocol implementation. We have thus adopted the pipe measurement strategy (referred to as PIPE) used in [4], using throughput as the primary performance index. The PIPE experiments measure the throughput between two communicating processes that take turns to do reads and writes through two shared UNIX pipes. Throughput is measured in bytes per second as a function of the number of bytes in each read/write system call. We call this benchmark program SPIPE, i.e., stream-based PIPE.

We investigate the throughput of the SPIPE benchmark by varying the following factors :

a. message size for each read/write system call.

* UNIX is a registered trademark of AT&T.

b. pass-through put procedure call versus service scheduling.

c. number of modules pushed onto the stream device.

d. null service module versus CRC-16 generation/verification service module.

2. OVERVIEW OF THE EXPERIMENT

The SPIPE benchmark creates one or more stream-based pipes (SPIPEs); each consists of a pair of duplex connections between two processes as depicted in Figure 1. In order to enable measurements on a stand-alone machine, a loopback pseudo-device at the bottom of the stream device is used to "connect" these two processes. Two loopback drivers are used for each pair of processes in one SPIPE instance. One process in the SPIPE connection writes a given number of bytes to one stream device and waits to read from the other stream device. At the same time, the other process attempts to read the given number of bytes from the first device, and then writes that number of bytes to the second device. Two processes are thus synchronized to do read and write in turns. There is a "forced" context switch after each write for both processes.

Since the methodology of the SPIPE and PIPE benchmarks are similar, we compare the throughput between these two measurements. The SPIPE uses no intermediate modules in the loopback driver and messages are passed down (and up) directly when the high watermark of each module is not exceeded†. The number of bytes transferred between these two processes divided by the elapsed time (in seconds) is the throughput of the SPIPE.

All measurements were conducted on a AT&T microcomputer†† for various message sizes : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4096 bytes. Measurements with only one SPIPE instance are reported here. Results with more than one SPIPE instance operating concurrently do not provide more insight in the context of the experiment.

The stream mechanism permits message processing and passing to be done in two ways depending on the urgency of the task to be carried out. Using pass-through procedure call facilitates the processing and speeds up the message flow from one module to another. In contrast, invoking service scheduling alleviates the contention for important resources such as CPU, but the system overhead in doing such scheduling can be significant. As part of our characterization, we compare the throughput between the pass-through put procedure call and the stream service scheduling strategy. We pushed one and two modules, a null put procedure and a null service procedure, on the stream device for these experiments. These null procedures only provide the necessary stubs to get their minimal functionality with no other processing involved. To ensure that the high/low watermarks do not affect the measurements, we used a high watermark of 10000 and low watermark of 1 for these experiments.

We also experimented with somewhat realistic modules that get pushed on the stream devices. These stream modules generate and append a CRC-16 (Cyclic Redundancy Check) checksum [5] for each message transmitted to the bottom level pseudo device driver. For messages received, the modules verify the correctness of the CRC-16 checksum and strip off the checksum bytes before passing the message back up. The CRC-16 checksum generation and verification processing is table-driven. For each byte received, two (one for each 4-bit nibble) look-ups were done in a 16 by 16 table. Using these modules, we have been able to gain some insight on the extra overhead of a modest amount of message processing relative to that of the basic stream mechanism. Checksum generation and verification is usually an integral part of any driver or protocol implementation that involves sending and receiving messages.

3. ANALYSIS OF THE RESULTS

3.1 SPIPE versus PIPE

Figure 2 depicts the throughput results for SPIPE, SPIPEB, and PIPE as a function of message size. SPIPEB refers to an SPIPE with QBIGB (BIG Queue Buffer) flag set to overwrite the default internal fragmentation for large messages. When QBIGB is set, the write buffer will not be fragmented at the default 256-byte boundary and a larger size (power of 2 up to 4096-byte) buffer can be used. However, unused buffer space with QBIGB set may become a problem in terms of efficient usage of shared buffers managed by the stream mechanism.

With message size of 1 byte, we can compute that it takes **4.90** msec of CPU time to pass down and back up one byte through the stream mechanism. This number derives from the inverse of the throughput result, and includes a *read*, a *write*, and a full *context switch* overhead. Figure 3 shows the percentage difference using the PIPE throughputs as the reference base line data. The SPIPE has 5% higher throughput for message size of 1 byte, and the gap narrows percentage-wise as message size increases. SPIPE throughput becomes worse when message size is greater than 256 bytes. This situation is reversed by

† The high and low watermarks, 10000 and 1, used in the experiment are different from those supplied with the original modules, i.e., 512 and 256.

†† This performance study was based on an early prototype of the Stream implementation; some changes and enhancements were made to the prototype since.

setting the QBIGB flag in the immediate module downstream as shown clearly in Figure 3. Large message size in SPIPEB provides the advantage of single data copy operation as compared to PIPE which copies data 1K bytes a time. Although setting QBIGB has significant impact on this benchmark, it should not be regarded as a universal rule in using the stream mechanism in general.

3.2 Pass-through versus Service Scheduling

Next we investigate the throughput differences between the pass-through put procedure call and service scheduling strategies for every message passed between modules. In order to ensure that exactly one service is scheduled for each message passing through, a high watermark of 10000 and a low watermark of 1 are used. Figure 4 depicts the percentage degradation in throughput relative to the zero-module pass-through SPIPE. It shows that service scheduling causes drops in throughputs of 22% and 33% relative to the pass-through put procedure call with one and two intermediate modules pushed on the stream, respectively.

3.3 CRC-16 Service versus Null Service

A null module was modified to perform table-driven CRC-16 processing on a per message basis for messages going downstream as well as upstream. Figure 5 shows the throughput for one and two CRC-16 modules with service scheduling. It is interesting to observe that throughput for the SPIPE benchmark starts to level off when message size is larger than 512 bytes. This implies that the per-byte overhead of the CRC-16 processing for each message becomes the dominating factor of the SPIPE throughput; thus, the relative overhead of the basic stream mechanism is small. For comparison, Figure 6 depicts the throughputs for the similarly configured SPIPE with null service and CRC-16 services for each message processed.

It is of interest to compute the per byte cost by inverting the throughput results. We can then verify the relative overheads between a null service module and a CRC-16 service module. The per byte cost (CPU time in microseconds) for message size of 1024 bytes for the null service module and CRC-16 module are listed below :

	one module	two modules
null service	23	30
CRC-16 service	106	193

For the one module case, the CRC-16 processing per module (2 queues) is about 83 microseconds per byte, and which is more than three times larger than the 23 microseconds per byte required for null service.

4. CONCLUSIONS

We have presented a simple characterization of the basic stream mechanism by using the SPIPE (stream-based PIPE) benchmark. The SPIPE throughput results are slightly better than those of the PIPE results when the message size is small. Relative throughput becomes worse as message size increases beyond 256 bytes. This is due to the internal message fragmentation in the stream mechanism. When SPIPE has the QBIGB set, it significantly outperforms PIPE with message size greater than 1024 bytes. When very little processing is done on the data, overall performance is heavily dependent on buffer sizes.

We have further quantified in the context of the SPIPE benchmark the throughput difference between direct pass-through put procedure call and service scheduling provided by the stream mechanism. The difference is significant enough to justify careful examination of more efficient implementation of service scheduling in the stream mechanism. It also indicates that we should be judicious in using service scheduling strategies in actual driver or protocol implementations.

Finally, the experiments with CRC-16 modules have shown that the overhead of the basic stream mechanism is relatively small for realistic implementations of protocols and drivers.

REFERENCES :

[1] D. M. Ritchie, "A Stream Input-Output System," AT&T Bell Laboratories Technical Journal, Vol.63, No.8 Part 2, October 1984, pp. 1897-1910.

[2] "UNIX System V : STREAMS Primer," AT&T Select Code 307-229, 1986.

[3] "UNIX System V : STREAMS Programmer's Guide," AT&T Select Code 307-227, 1986.

[4] J. P. Lankford, "UNIX System V and 4BSD Performance," 1984 Summer USENIX Conference Proceedings, Salt Lake City, June 1984.

[5] J. E. McNamara, "Technical Aspects of Data Communication," Second Edition, Digital Press, Bedford, Mass., 1982.

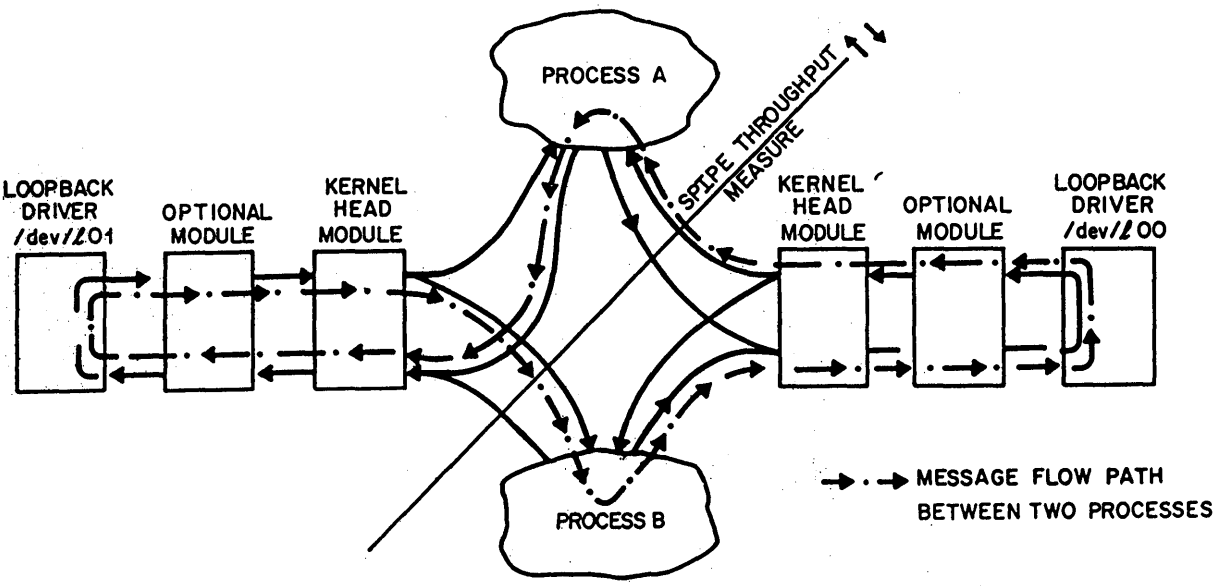


FIGURE 1 TWO COMMUNICATING SPIPE PROCESSES

Figure 2. SPIPE(B) and PIPE Throughputs

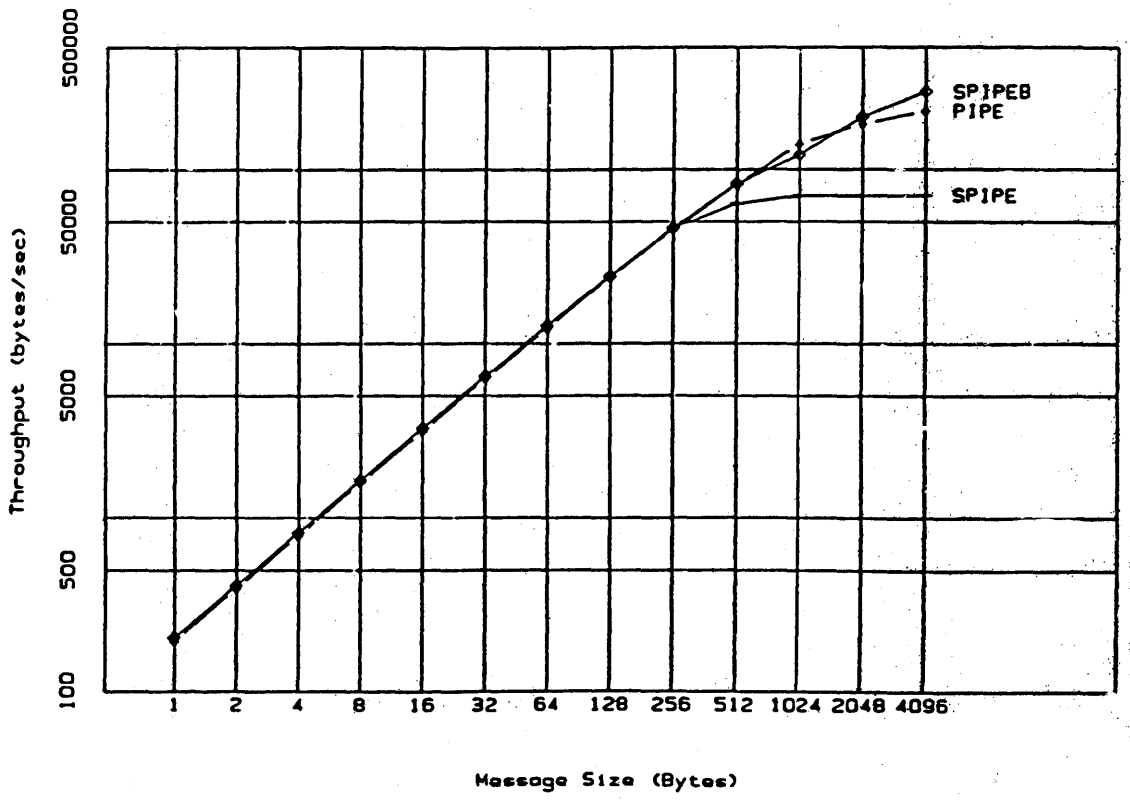


Figure 3. Throughput Comparison : SPIPE(B) versus PIPE

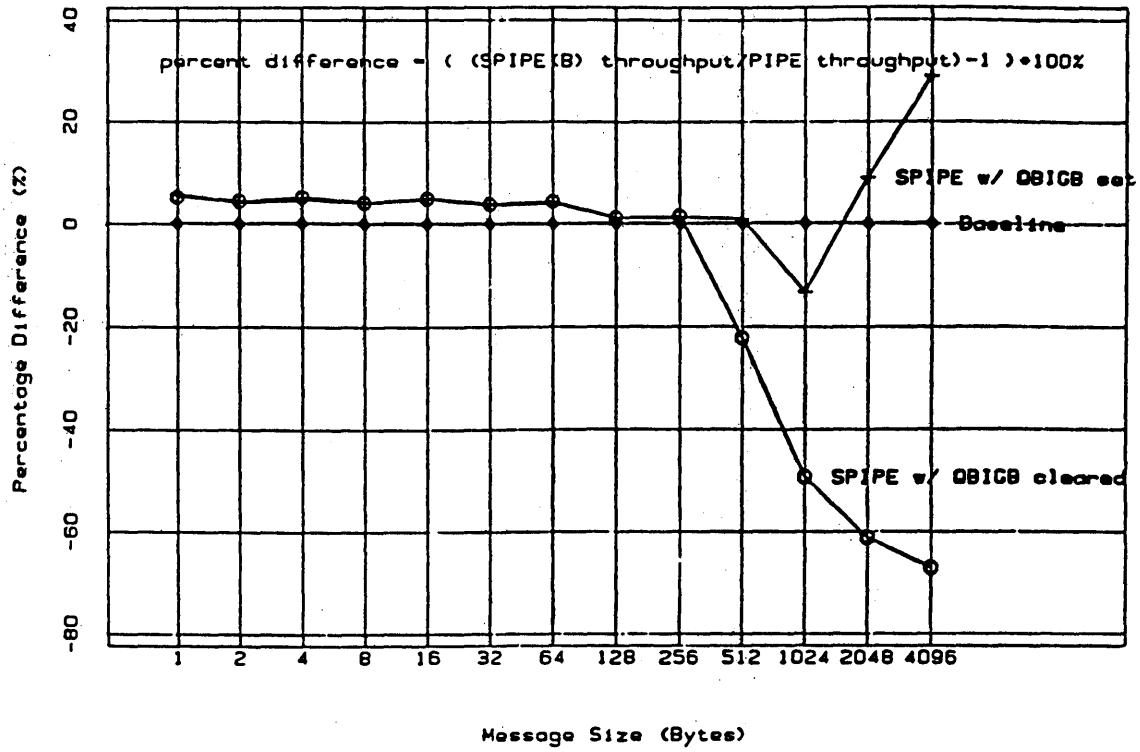


Figure 4. Degradation Due to Service Scheduling

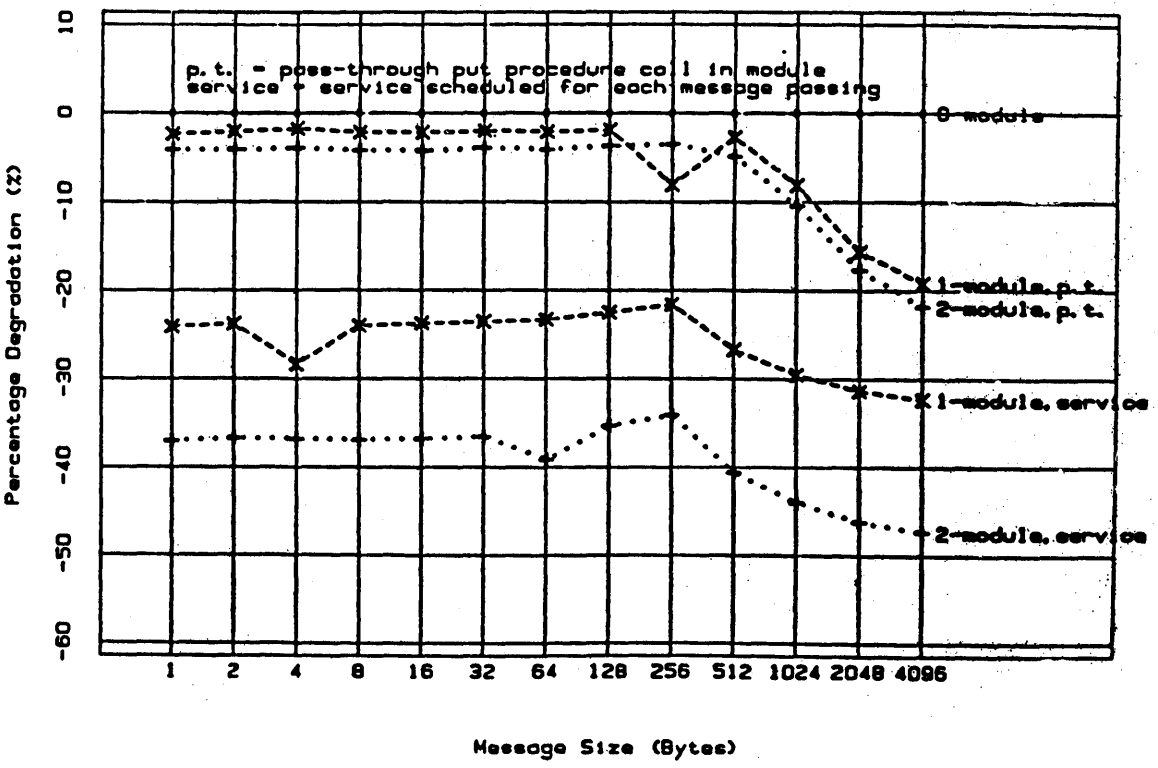


Figure 5. SPIPE Throughput with CRC-16 Modules

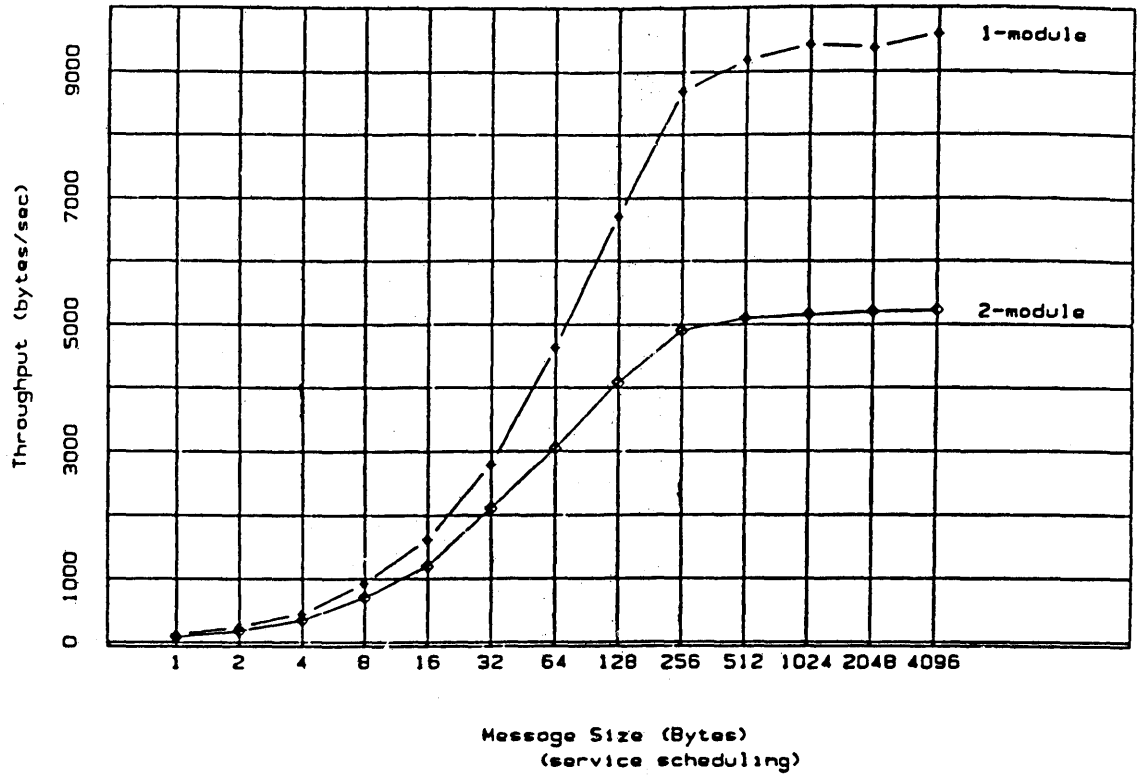
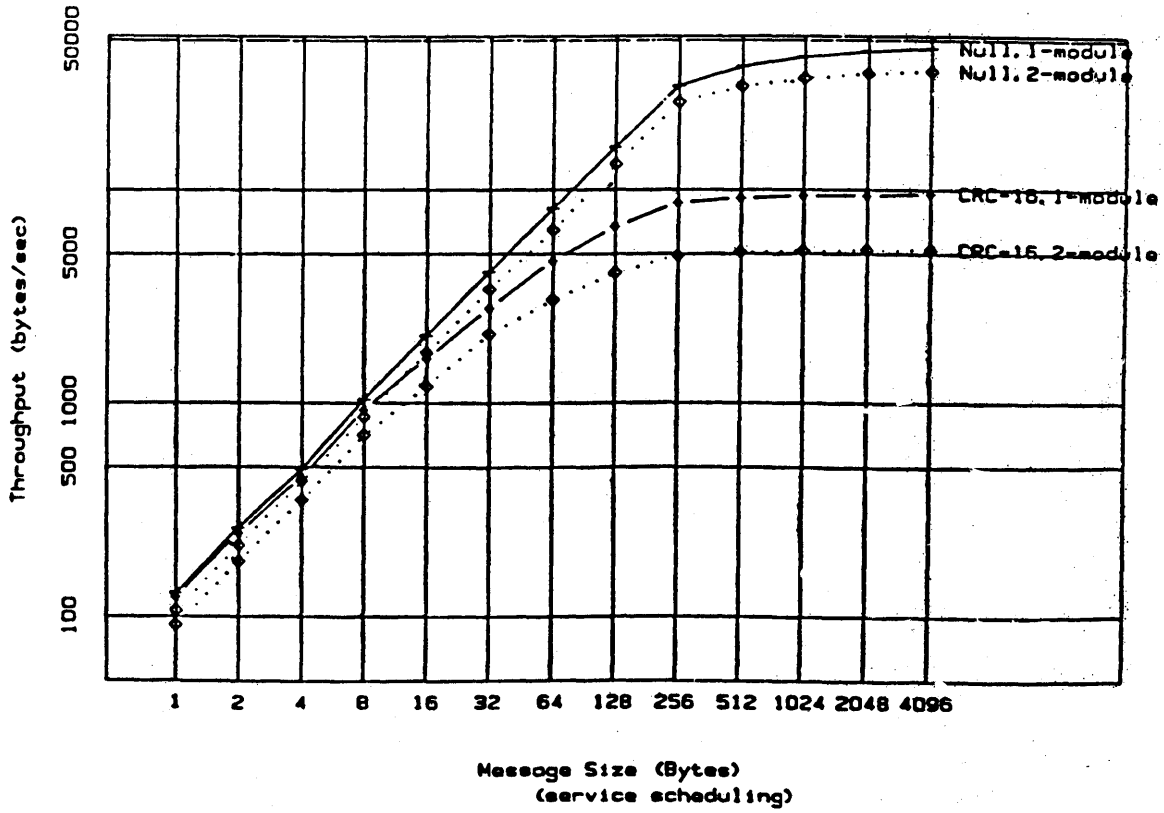

1-module
2-module
Throughput (bytes/sec)
0 1000 3000 5000 7000 9000
1 2 4 8 16 32 64 128 256 512 1024 2048 4096
Message Size (Bytes)
(service scheduling)

Figure 6. SPIPE Throughputs : Null versus CRC-16



ANALYZING FILE SYSTEM FRAGMENTATION ON UNIX® SYSTEMS

Behrokh Samadi
AT&T Bell Laboratories
Crawfords Corner Road.
Holmdel, New Jersey 07733

ABSTRACT

A model for analyzing the extent of fragmentation in a UNIX* System V file system is described in this paper. The extent of fragmentation is measured in terms of the reductions in disk access time if the file system is reorganized. Since reorganizing a file system is a time consuming operation, such a measure provides an incentive to reorganize the file system if noticeable performance improvement is expected to result. In this paper we describe the model, its implementation and some experimental results in controlled and real environments.

The original motivation for this model is an expert system for UNIX System tuning. The paper briefly describes this tuning tool and how the model is utilized by this tool to provide more specific information to the user.

1. INTRODUCTION

In this paper we describe a model for analyzing the extent of fragmentation in the physical layout of files on a UNIX System V file system. The amount of fragmentation is measured by the reductions in the disk i/o time if the file system were optimally organized. The fragmentation results from frequent addition and deletion of file blocks in a multi-user environment, and it can introduce excessive disk head movements in sequential file access. This methodology has been implemented and the implemented module provides information regarding the organization of file systems to TUNEX[1], an expert system for UNIX tuning[1] Here, we present the model, its implementation and a number of experimental results. This model is based on the UNIX System V file systems and its input are derived from the System V measurement utilities. Thus, all references to UNIX system refer to System V version.

The System V file system construction command (*mkfs*(1M)[2]) [2] uses two parameters, *gap* and *blocks/cylinder*, to arrange the list of free blocks in an order that typically results in more efficient access time. The parameter *gap* is the number of physical blocks between every pair of consecutive blocks on the free block list, and *blocks/cylinder* is the number of blocks on a cylinder. With proper selection of *gap*, it is expected that initially the consecutive file blocks are within some optimal rotational distance on the same cylinder if possible. The idea is to reduce the disk seek and latency times by placing the files in a logically contiguous space.

The operating system does not make any attempts to maintain this organization however. As file blocks are added and removed, they become more scattered over the file system space. Hence the file system's layout starts to deteriorate that results in excessive seek and latency times. This is particularly significant since the positioning of the disk head over the requested block is in general the most time consuming of the i/o process.

The age of a file system can also reduce the access efficiency of its directory blocks. Directory blocks hold an entry for every file in the directory. As files are deleted, the corresponding entries in the directory blocks are nulled out. Although these null entries may eventually be reused, their existence in excessive numbers, can cause long directory searches and i/o overhead.

A UNIX system utility (*dcopy*(1M)) reorganizes the file system by placing consecutive blocks of a file within an optimal rotational gap, on the same cylinder if possible. It also places the more recently accessed files closer together. In addition, it compacts the directory blocks to remove the null entries, thereby reducing directory sizes.

The reorganization is a time consuming operation and the file system must be unmounted, otherwise, its consistency cannot be ensured. Furthermore, there is

---

* UNIX is a registered trademark of AT&T

1. TUNEX is available through AT&T UNIX System Toolchest.
2. Referring to command names in the UNIX System V reference manuals.

little indication of how unorganized the file system is and how much its reorganization would affect the performance. In very dynamic environments, an organized file system can become unorganized in a matter of days. So, the performance gains resulting from a reorganization may not last long enough to justify the efforts needed to perform this operation. For these reasons and because the improvements are difficult to quantify, system administrators either hesitate to reorganize file systems, in which case the disk i/o operations may become very inefficient, or they reorganize regularly, which may not be necessary.

Here, a model is described that evaluates the extent of randomness in the physical layout of files. The results can help us make a decision on whether a reorganization is necessary and worthwhile. The model includes a number of assumptions and approximations that may at first seem rather crude. Nevertheless, we are able to show that this simple model is sufficiently accurate to distinguish situations where the gain from reorganization will or will not be significant.

The paper is organized in 5 sections. In section 2 the details of the model are presented. This section also describes the approach in obtaining values for model parameters. Section 3 contains the experimental results. Section 4 describes the original motivation for the development of such a model which is an expert system for UNIX system tuning. Finally, in section 5 a summary of the paper is presented.

2. THE MODEL

The fragmentation measure, C, is defined as the average reduction in disk access time per block access, if the file system is reorganized. This is equal to the unnecessary disk head movements resulting from the fragmented layout. This movement is called *unnecessary* because it is not required when the file system is organized. Thus,

$$C = S+L$$

where S and L are the average unnecessary seek and latency times per block access.

The random positioning of file blocks on the disk drive affects i/o performance most when consecutive file blocks are accessed consecutively. This type of access will be called *sequential file access*. Many commands and applications require sequential access to data and executable files. In the case of sequential file access, the unnecessary seek and latency times correspond to the seek and latency times between accesses to consecutive blocks of files. We make the assumption that:

(A1) The application programs access files sequentially.

This assumption can be relaxed by letting a percentage of file i/o activities to be of sequential nature and scale the performance measure appropriately.

In the presence of other disk i/o requests and depending on the disk scheduling policy, two consecutive i/o requests from the same application are not necessarily serviced consecutively. However, with a number of elevator type [3] and shortest seek time first disk scheduling policy, there is a high probability that consecutive i/o requests to the same or neighboring cylinders are served consecutively. On the other hand, if the requested blocks are far apart, the driver may schedule other requests between the two. In this case, if the file blocks are scattered, the disk head needs to make some unnecessary movements to read the next requested block in sequential file access. To compute the unnecessary disk head movements, we further make the assumptions that:

(A2) If two consecutive file blocks $x-1$ and x are on the same or adjacent cylinder, requests to access these two blocks are serviced consecutively. Otherwise, after accessing block $x-1$, the disk head may move to service block x from any other cylinder i, with a probability distribution independent of blocks x or $x-1$ and proportional to the long term frequency of access to cylinder i.

The second part of this assumption is rather pessimistic due to the locality property of access requests and the use of elevator strategies. Now, let:

p_x = *the probability of an access request being for block x*
i_x = *the cylinder on which block x is located*
r_i = *the probability of an access request being for cylinder i*
$s(i,j)$ = *avg. seek time to cylinder j given the disk head is on cylinder i*
$h(x)$ = *avg. unnecessary seek time to access block x*

Then

$$S = \sum_{\text{all files } f} \sum_{\text{all blocks } x \in f} p_x h(x)$$

Following assumptions (A1) and (A2),

$$h(x) = 0 \qquad \text{if preceding logical block x-1 is on the same or adjacent cylinder}$$

$$= \sum_{\text{all cylinders } i} r_i\, s(i,i_x) \qquad \text{otherwise}$$

The density functions p_x and r_i are derived from the measurement data by making the assumption that

(A3) r_i is independent of the previous accesses and is equal to the frequency of access to cylinder i obtained from the long term measurements. p_x is also assumed to be independent of the previous access and is derived from r_i by assuming that the allocated blocks within a cylinder have an equal probability of being accessed.

The seek time between two cylinders i_y and i_x, $s(i_y,i_x)$, is assumed to be a function of d, the number of cylinders (distance) between the two. The following two formulae were used to evaluate $s(i_y,i_x)$, the average seek time.

(A4)

$$s_1(i_y,i_x) = a_1+b_1*d$$

$$s_2(i_y,i_x) = a_2+b_2*\sqrt{d}$$

where

$$d = abs(i_x-i_y)$$
$$= \textit{the seek distance (in cylinders) between two cylinders } i_x \text{and} i_y$$

The values for a_k and b_k, k=1,2, are computed in such a way that they satisfy the maximum (entire volume) and minimum (one cylinder) seek time values as reported in the disk drive's manual. In one seek time evaluation function, the disk head moves with constant velocity across cylinders. In the other, it moves with constant acceleration. Our experiments showed that the final results are not too sensitive to the choice of the formula.

To evaluate the unnecessary latency per accessed block, L, we have (from (A1, A2, A3)):

$$L = \sum_{all\ x} p_x\ l(x-1,x)$$

where $l(x-1,x)$ is the latency to access logical block x given preceding logical block $x-1$ was accessed last. Letting blk(x) to be the physical block number corresponding to x,

$$l(x-1,x) = 0 \quad \textit{if } x \textit{ is the first block of the file}$$

$$= ((blk(x-1)\oplus gap)\ominus blk(x))* \textit{(rotational time over one block)}$$
$$\textit{otherwise}$$

Here $x\oplus y$ and $x\ominus y$ are defined as $(x+y)\ mod\ (blocks/cylinder)$ and $(x-y+blocks/cylinder)\ mod\ (blocks/cylinder)$ respectively. *Gap* is the optimal rotational distance between two consecutive file blocks, and *blocks/cylinder* is the number of blocks on a cylinder. These two values are parameters of the file system construction and reorganization commands and depend on the type of the disk drive.

Once C is computed, it is used to predict disk utilization after reorganization. For this, the measurement data from the system activity report (*sar(1M)*) is used. This command reports measured disk utilization (ρ) and computed average service time ($\bar{x}$) during a measurement period. The approximate disk utilization after reorganization (ρ'), is obtained from:

$$\rho' = \rho\frac{\bar{x}'}{\bar{x}}$$

where

ρ = *disk utilization before reorganization*
$\bar{x}$ = *average service time before reorganization*
$\bar{x}' = \bar{x}-Cq$ = *average service time after reorganization*
q = *ratio of file system i/o to the total disk i/o*

2.1 MODEL PARAMETERS

The model parameters need to be measured or extracted from the system measurement data. In this section it is shown how these parameter are obtained and the assumptions that have to be made because of the unavailability of more accurate data.

To obtain the seek distance in the number of cylinders, and the latency in the number of blocks, a program has been developed that reads all file block numbers and produces a mapping[3] from the file blocks to the disk blocks. In addition, it counts the number of allocated blocks on every cylinder. This count is used to distribute the disk profile data to finer granularity (explained below). The program also looks at the directory blocks to measure the number of directory blocks that will be saved as a result of reorganization.

The cylinder and block access frequencies are obtained from the System Activity Disk Profile (*sadp*(1M)) data. This utility samples the number of accesses to 8-cylinder groups. It has been assumed that:

(A5) A cylinder on any 8-cylinder group is accessed proportional to the number of allocated blocks on that cylinder. Furthermore, if there is an

3. The file headers (*inodes*) provide file system block numbers for all blocks in the file. These numbers can be mapped to physical block locations by considering file system block size and the starting block of the disk partition on which the file system is located

access request for a cylinder, the blocks within that cylinder have an equal probability of being accessed.

There is an exception to this rule, and that is in the case where the boundary of two file systems lies inside an 8-cylinder group. Typically, the cylinders at the beginning of the file system have higher access frequency than those at the end. Hence,

(A6) Each subgroup on a boundary 8-cylinder group is first assigned a part of the disk activity profile proportional to the disk profile of its adjacent 8-cylinder group. Then, within each subgroup, the profile is distributed according to (A5).

The modules for data preparation and model evaluation are packaged together into a utility command that uses the system measurement data and produces the predictions. The run time of the program depends on the size of the file system. It could be a time consuming operation depending on how many blocks are allocated. On a file system of size 30K blocks, 60% full, the total cpu time on a 1 MIPS machine is about 6 seconds.

3. EXPERIMENTAL RESULTS

Experiments were conducted to verify the validity of the model. Note that the model included a number of assumptions and approximations that are rather coarse. However, we are interested in predictions that are accurate enough that can differentiate between a badly organized file system and one that is not too fragmented. The experimental results show that the predictions can provide us with some insight into the effectiveness of reorganization. In this section the results of two experiments in controlled and real environments are presented.

The first experiment consists of running an i/o-intensive benchmark on a dedicated machine - one with no interfering workload. The benchmark's i/o requests are mainly directed to an unorganized file system . This file system was fragmented by running the reorganization utility with wrong values for *gap* and *blocks/cylinder*. The system activity measurement (*sar*) and the disk activity profile (*sadp*) during the benchmark period were recorded.[4] Using the data from the run, the model was used to predict the disk utilization after reorganization. The result is given in Figure 1.

In this figure, *col_day* and *col_time* are the date and time when the data (disk activity profile) was collected. *Eval_day* and *eval_time* are the date and time of evaluation of the file system. *Cq*, the reductions in disk service time, is given in milliseconds. ρ is the measured disk utilization reported at *col_time* by *sar*, and ρ' is the predicted utilization after reorganization. Finally, *nullblk* is the total number of directory blocks saved as a result of reorganizations.

The file system is then reorganized using *dcopy*. The same benchmark is run on the two file systems. The *sar* results from both the first and second benchmark runs appear in Figure 2. In this figure *%busy* is the measured disk utilization (ρ), *avque* is the average queue length given the disk drive is busy, *r+w/s* and *blks/s* give the number of i/o requests per second, *avwait* and *avserv* are the average waiting and service times on the disk drive respectively. The model predicted that the utilization will reduce to 51% after reorganization. The actual measurements after running the same benchmark on the organized file system (Figure 2) show that the disk utilization reduced to 48%.

Finally, in Figure 3 we show the results of running the model with the data collected when the benchmark was running on the organized file system. Minimal performance improvements are expected if the file system is reorganized.

The second experiment was conducted on a real system. The file system under study was naturally fragmented. That is, unlike the previous model, its organization was not modified for the purpose of the experiment. A similar benchmark was run on this system and the system measurement data was collected. Note that we had to run a benchmark, since we cannot compare the after and after measurements if the workloads are not similar. However, we did let the real workload interfere with our measurements.

The file system was then reorganized and the same benchmark was run on the system. The *sar* results

col_day	col_time	eval_day	eval_time	Cq	ρ'	ρ	nullblk
3/10/86	9:20:03	3/10/86	9:32:12	4.40	51	60	1

Figure 1. Predicted Disk Utilization after Reorganization.

4. The disk capacity in this experiment is 72 MB.

Before the change:

device	%busy	avque	r+w/s	blks/s	avwait	avserv
hdsk-0	60	7.2	20	40	184.9	29.7

After the change:

device	%busy	avque	r+w/s	blks/s	avwait	avserv
hdsk-0	48	5.0	20	40	96.3	24.1

Figure 2. Sar Results for the Disk Activities before and after the Reorganization.

col_day	col_time	eval_day	eval_time	Cq	ρ'	ρ	nullblk
3/10/86	12:40:02	3/10/86	13:46:08	0.36	47	48	0

Figure 3. Model's Predictions after Reorganization

before the change:

device	%busy	avque	r+w/s	blks/s	avwait	avserv
dsk-0	12	1.8	6	15	15.3	20.1
dsk-6	65	4.5	25	50	91.5	26.2
dsk-9	21	34.4	8	16	880.8	26.4

after the change:

device	%busy	avque	r+w/s	blks/s	avwait	avserv
dsk-0	5	1.6	2	11	14.3	23.5
dsk-6	43	6.6	23	46	105.7	18.8
dsk-9	17	1.9	7	13	22.7	26.3

Figure 4. Sar Results before and after Reorganization

col_day	col_time	eval_day	eval_time	Cq	ρ'	ρ	nullblk
10/28/86	21:03:29	10/28/86	21:33:17	7.45	46	65	0

Figure 5. Predicted Disk Utilization after Reorganizing a File System on dsk-6

from before and after the change are shown in Figure 4. The file system under study is placed on dsk-6[5]. The results of the prediction model are shown in Figure 5. The predicted utilization was 46%, while the measured utilization after the change was 43%. Note that in Figure 4 the average queue length at lower utilization is higher than that at higher utilization. This is because the average queue length is conditional on the disk being busy.

The difference between the predicted utilization and the measurements after the change is partly due to the fact that the number of i/o requests to the disk ($r+w/s$ column) are not the same during both benchmark runs. More i/o activities occurred during the first run of the benchmark. The model, therefore, based its knowledge on that i/o traffic, predicted a higher utilization. Nevertheless, the results are close enough that they validate the predictions.

4. RELATED ISSUES

The motivations for this model arose in the development of TUNEX, an expert system for UNIX system tuning. Initially TUNEX incorporated the system tuning and administration rules to find performance problems and recommend tuning solutions. However, it was soon found out that the rules are too general and in order to make quantitative recommendations, we need to perform quantitative analysis that need more specific information on the state of the system. This model is developed for TUNEX to ascertain whether i/o performance can be improved by reorganizing the file systems.

Once included in TUNEX, many other issues arise that need to be dealt with. For example, the quantity Cq (Figure 5) is a measure of how unorganized a file system is. However, ρ' tells us how effective the reorganization will be. Note that if ρ, the disk utilization, is small, no matter how unorganized the file system is, performance improvements will not be noticeable after reorganization. TUNEX recommends a reorganization when the disk utilization is above a threshold value and predicted utilization is below that value.

Another issue is the duration of time before the file system becomes unorganized. In a very dynamic environment an organized file system can become unorganized fast enough that a reorganization may not be cost effective. For this, TUNEX needs to track a history of the file system state, so that it can make a recommendation if a reorganization is worthwhile. This feature has not yet been implemented in TUNEX.

Given the predictive model, there is a question on which measurement data to use for the model. Data collected at different points in time result in different predictions about the extent of performance improvement. Organizing the same file system may be predicted to improve disk utilization by 5% or 10% depending on the time the input data was collected. TUNEX lists the range of predicted improvements resulting from the input data taken over periods of time when the disk drive was very busy (over 50% utilized).

5. SUMMARY

We presented a model for evaluating extent of fragmentation on a System V file system. Model's predictions in terms of performance gains help administrators make proper judgement on running a normally time consuming maintenance operation. We showed some experimental results that validate the model and discussed how this module was included and used by an automated tuning tool.

REFERENCES

[1] Samadi, B., "A Knowledge-based System for Performance Tuning of the UNIX Operating System,", USENIX Winter '87, Washington D.C.

[2] UNIX System V, "Administrator Reference Manual," AT&T.

[3] Teorey, T.J. and Pinkerton, T.B., "A Comparative Analysis of Disk Scheduling Policies," Comm. of the ACM, No. 3, Vol. 15, March 1972, pp. 177-184.

5. 675 MB disk drive.

CAPACITY ANALYSIS OF PBX SYSTEMS

T. L. LO and J. PENG
McDonnell Douglas
Aerospace Information Services Company

ABSTRACT

A private branch exchange (PBX) is an electronic switching system that provides connections for voice communications. Recent PBX technology development allows the integration of voice and data traffic on PBX systems without compromising the quality of communications. This paper describes several capacity analysis algorithms for determining proper processing capacity of a PBX system based on processing environment, requirements and growth. Two case studies are provided to demonstrate the applicability of those algorithms.

INTRODUCTION

The private branch exchange (PBX) technology has evolved to parallel the computer technology in size, complexity, and importance. For years data has been carried on voice lines via sophisticated voice networks which has evolved since the inception of the telephone. Recent introduction of digital PBX and integrated voice and data services are perceived as an economic alternation in networking. The digital PBX technology provides interface capability to local area networks (LANs) and office automation. Unfortunately, most of today's voice and data integration of networks is external to the PBX with 90% of the PBX's installed still utilizing voice-only capabilities.

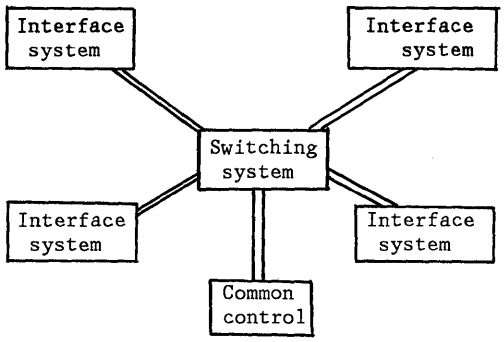
To select a proper PBX to accommodate present and future processing requirements is similar to selecting a computer system since capacity, service goals, cost and workload requirements all need to be considered. Only the evaluation criteria are different. For example, least-cost routing, speed dialing, call blocking probability and call handling capacity are the most popular PBX selection criteria whereas CPU MIPS, channel data rate, memory and cache size are pertinent parameters for computer selection.

This paper describes several algorithms for selecting a PBX system based on the requirements. The first section reviews fundamental properties of a modern PBX system. The second section discusses general PBX selection considerations and the third section presents several models to deal with performance prediction. Case studies are provided to illustrate the application of these models.

PBX ARCHITECTURE & TECHNOLOGY

PBX ARCHITECTURE

A PBX system is composed of three basic elements: common control, switching matrix and interface unit.



The common control governs the operation of the entire PBX system. It determines the allocation of internal resources for the function to be performed. It usually employs one or more microprocessors (CPUs) with random access memory (RAM) and read-only memory (ROM), backup disk or tape unit, and separate stand-alone software for call data recording and processing. The switching matrix provides the communication paths required for the interconnection of the interface units. It can be operated under three switching techniques: space division, time division and frequency division multiplexing, (SDM, FDM and TDM respectively). The speed of the switching matrix determines the traffic capacity of each PBX. The inclusion of data traffic impacts what is available to voice and alters the statistical behavior of the switch. The interface unit normally consists of "lines" from users to switching matrix (line cards) and "trunks" to outside switches (Trunk cards). The unit interfaces with all external equipment and is modularized to accommodate different line size requirements. The inclusion of protocol converters and LAN interfaces, as well as high speed trunks, make the interface unit more adapted to support distributed processing.

PBX SWITCHING TECHNOLOGY

The basic PBX switching technique deals with signal transmission in two distinct types: analog and digital. Analog transmission means that the signal transmitted is a direct analog of the actual signal. For example, in a voice signalling, an analog signal looks like a continuous wave which is modified as the voice frequency changes. In other words, analog signal changes in amplitude, increases in frequency, or drops to zero, depending upon the voice frequency gets louder, higher or stops. Digital signals are not continuously variables. They exist as a series of pulses, each represents one of a finite number of values at any instant.

The major difference between analog and digital signalling, therefore, is in their means of transmission. Analog signals are amplified as they are transmitted and amplification usually does not distinguish between the original signal and noise which may have cropped up on the line during the transmission. Digital signals do not have to be amplified, but are generated such that new signals are created, based on the measured values, to simulate the old ones. Thus all external noises are dropped during the regeneration.

A true digital switch PBX is a PBX with an internal switching matrix that switches voice (analog) and data (digital) signals in digital form. But a "computerized" PBX only implies the digital common control, not necessarily the switching matrix. Therefore, an "automated time division multiplexing PBX" may very well switch analog signals. PBX technology, especially switching technique, evolves following VLSI and ULSI development trends. PBX technology can generally be categorized into four development generations:

- First generation (1960's) - electronic and mechanical hard-wired logic-based switches using diodes and transistors. It was totally analog transmission with time-division multiplexing, but not computerized.

- Second generation (late 1960's -early 1970's) - primarily analog transmission with space division multiplexing, but with some digital capabilities requiring the assistance from external devices such as modems. It is microprocessor based and features stored program intelligence such as least-cost routing, security control, and automatic distribution of calls.

- 2.5 generation (late 1970's) - second generation PBXs with enhancements to accommodate data switching is the key feature of this half generation PBX development. PBXs utilize analog signalling between phone handset and PBX; data signals are converted to analog signals at the switch by a coder-decoder (CODEC) facility.

- Third generation (early 1980's) - most PBXs were digital-based to integrate voice and data at the phone handset rather than at the switch matrix. The switching technique utilizes pulse-code modulation (PCM) digital time division multiplexing and is characterized by "distributed" stored program intelligence.

- 3.5 generation (mid 1980's) - Several third generation PBXs with cabled-based local area networks (LANs) capability can be considered as one step closer toward the fourth generation PBX technology. In addition to the LAN interface, these PBXs can dynamically switch traffic using high speed bandwidth, such as T1 and satellite circuits, as well as the support of X.25 packet switching network interface.

- Fourth generation (late 1980's - early 1990's) - fiber optics and microwave links will be widely used to provide very large bandwidth (e.g. 500 mbps) for every workstation and each station will have the capability to perform teleconferencing, file manipulation, and complex graphics.

PBX CAPACITY ANALYSIS CONSIDERATIONS

SWITCHING CAPACITY

The most common PBX capacity indicator is "one hundred (Roman numeral C) Call Second, or CCS." CCS capacity is similar to MIPS (millions instructions per second) as it relates to computer processing capacity. Therefore, CCS can be used as a common denominator to provide some indication of performance capacity for each PBX. For example, 36 CCS can be represented as the capacity per hour per path, derived by 60 minutes X 60 second divided by 100. Most PBXs require a 6 CCS per call as a minimum capacity for a normal voice system. If data is introduced through the PBX, 24 CCS per call would be required to statistically handle the traffic since a data call has approximately four times more impact on PBX capacity than a voice call. A typical mix of voice/data in today's PBX environment is 75% voice and 25 % data (OPDE85).

To determine a particular PBX's capacity, the CCS can be interpreted in terms of number of simultaneous calls per hour. For example, AT & T System 75 PBX has a rated 8,676 CCS capacity. A worst case scenario of 36 CCS per call with full-time data use would offer 241 simultaneous calls per hour. A best case with 6 CCS per call, with voice only requirement, would support 1,446 calls per hour capacity. If we use a typical 75/25 voice/data mixed workload, the System 75 capacity can approximately support 600 voice only and 200 voice/data stations (600 + 4*200 < 1,446).

Although CCS is a common PBX capacity indicator, it is only one of many variables which must be considered when selecting a PBX for a particular processing environment.

TRAFFIC STATISTICS

Since the majority of today's PBXs are voice-based, voice traffic statistics, such as number of distribution of incoming and outgoing calls, number of long distance calls, and call duration, are prominent. An average phone user's profile can be derived from these statistics, which can also be used to determine future usages.

In addition to traffic statistics, the probability of call blocking is also of importance. That is, when a user needs to access the phone, there is a chance that he/she will not be able to use the phone due to all-line-busy condition. A probability of 0 means a non-blocking capacity. The U.S. telephone industry uses a block probability of 0.01 (one out of every 100 calls is likely to be blocked) whereas a 0.005 is used by European telephone industry.

The probability of call blocking is indirectly affected by the number of trunk lines that a PBX can support. The trunk lines can be grouped via a TIE, FX, WATS or DOD/DID line. A TIE line is a leased line connects two PBXs without having to dial normal phone number. A FX (foreign exchange) line provides telephone service from a central office (CO) which is outside the caller's exchange area. A caller may, in the simplest form, dial one or two digits to obtain a dial tone for subsequent calls. A Wide Area Telecommunication Service (WATS) line allows calls to specified geographic areas with potential lower cost long distance services. WATS lines essentially come in two types: INWATS and OUTWATS. The Direct Outward Dialing (DOD) and Direct

Inward Dialing (DID) lines allow callers to dial an extension without the need for assistance from the PBX attendant. They are connected via PBX to local COs.

COMMUNICATIONS REQUIREMENTS

The third factor for determining a PBX capacity is the voice communication requirements. The most desirable features are automatic route selection, signal transmission, centralized attendant, traffic statistics recording, voice/data integration, maintenance and control, and user programmable features.

The automatic route selection (ARS) feature allows a PBX to determine the route for calls as THE user begins to dial the first few digits. Queuing should be considered as part of ARS. There are generally two types of queuing: off-hook and call-back. An off-hook queuing calculates the average time that a caller must wait, in an all-line-busy condition, for the first trunk line to become free. This queuing time should not be more than 10 seconds. A call-back queuing allows a caller to place himself in a queue and hang up. The PBX system will inform the caller within a specified time interval, normally less than 10 minutes.

Signal transmission feature requires a PBX to switch signals on trunk lines to avoid unnecessary turnaround delay especially in a TIE-trunk network which connects two PBXs with dedicated lines for fast communications between the two PBXs. Since the analog signaling will probably be the voice communication standard for a long time, a connection from an originating to a terminating PBX should follow the "VLN+4" rule. This rule introduces "loss" into a line to reduce the echo and the loss required is called Via Net Loss (VNL). In order to make echo level low enough, an extra amount of loss, approximately 4 decibels (dB) is required, divided up to 2 dB in each end of the line.

Centralized attendant - short TIE trunks which are used to distribute incoming calls from the attendant switchboard to nearby PBXs, are inexpensive and the swap of TIE trunk cost for attendant salary makes economic sense. Furthermore, desk-to-desk dialing is offered by most PBXs, thus reducing the dependency on switchboard attendants.

Traffic statistics recording - most PBXs provide the call dial recording (CDR) capability. In addition, the dialing and ringing times, non-productive intervals and aborts, and queuing time and length are desirable statistics. For capacity planning purpose, queuing, trunk call distribution, and processor utilization are pertinent information to record. Once the information is collected, PBXs should provide the post-processing functions to report system resource usages and calling statistics.

Voice/data integration - the primary motivation for integrating voice and data is to reduce switching and transmission costs, consolidate facilities, and improve administrative efficiency. The developments of local area network technology, intercity digital switching (e.g. integrated services digital network), and voice/data workstations all make voice/data integration affordable and attractive.

Maintenance and control - one of the most frustrating parts of telephone system management is getting changes and additions made. The PBX internal system should be able to display information such as which phone is on which pick-up group, who hunts to whom, which users are assigned to which extensions, and which phone sets are assigned to what switch matrix input.

User friendly features - most modern PBXs provide users with programmable features through the stored-program control intelligence. Features such as call forwarding, ring again, night attendant, call waiting indication, and inter-desk dialing, are the most welcome ones.

MATHEMATICAL MODELS FOR PBX CAPACITY ANALYSIS

ERLANG-B MODEL

The common approach to analyze PBX capacity is to use analytic modeling. These models are of particular interest: Poisson, Erlang-B, and Molina. All three deal with three types of input variables: number of trunk lines, traffic volume, and probability of call blocking. Knowing the values of any two, the value of the third one can be readily determined. Since Erlang-B model is the most widely used model in telephone industry, it is discussed here.

The "number of line" variable implies the number of trunk lines that a PBX needs to support. The "traffic volume" variable requires two types of input information: average number of calls per unit of time per phone, and average call duration which is defined as the elapsed time between phone handset off- and on-hook. The product of these two is called an Erlang. That is,

$$\text{Erlang} = (\#\text{ of call/time/phone}) * (\text{call duration/phone})$$

The probability of call blocking indicates the chance that a phone call is blocked because of all-line-busy condition. The general equation to determine the probability of blocking, assuming the number of trunk line and traffic volume are given, is based on recursive relations,

$$P(n) = (P(n-1) * A)/(I + P(n-1) * A)$$

where $I = 1,\ldots,T$ (T is number of trunk line)
$n = 2,\ldots, T+1$ (representing the recursive relation)
A = average traffic volume in Erlang
$P(1) = 1$ is the starting value

For example, if A is 4.51 Erlangs and T = 6 lines, then the probabilities of blocking would be:

X	P(X)
1	1.0
2	0.8185
3	0.6486
4	0.4937
5	0.3576
6	0.2439
7	0.1549

The result can be interpreted as follows. Given 6 trunk lines with the average traffic volume of 4.51 Erlangs, the probability of blocking any access to trunk line is 15.49%; or roughly one out of every six accesses is likely to be blocked due to trunk line shortage. The general practice of finding a desirable probability of blocking is to increase the number of trunk lines with traffic volume held as constants until the desirable probabilities are found.

MULTISERVER QUEUING MODEL

In a telephone information inquiry system, the number of attendants is of primary concern since it determines how long a caller must wait for an available attendant if the call can not be handled right away. The multi-server queuing models (MART72) can be used to determine the proper number of attendants required to meet the minimum service goal.

The model assumes the following:

* the system is in steady-state condition.
* incoming calls arrive in a random pattern.
* the service time (the length of time that an attendant spends servicing an inquiry) is independent of the service time of previous inquiry.
* all attendants are capable of providing the same type of service.
* the service policy is first-come-first-served.

Let

U = server utilization
N = average arrival rate
Ts = average service time
M = number of servers in parallel on a multiserver system
Tq = average queuing time
Q = average queue length
Tt = total time spent in the system
B = probability that all servers are busy
T = total number of inquires in the system

Therefore,

$$U = N * Ts/M$$
$$B = X/Y$$
$$X = 1 - K$$
$$Y = 1 - U*K$$

$$K = \sum_{N=0}^{M-1} \frac{(MU)^N}{N!} \Big/ \sum_{N=0}^{M} \frac{(MU)^N}{N!}$$

$$Q = B * (U/(1 - U))$$
$$T = B * (U/(1 - U)) + M*U$$

$$Tq = \frac{B * Ts}{M(1 - U)}$$

$$Tt = \frac{B * Ts}{M(1 - U)} + Ts$$

CASE STUDIES

Two case studies are presented here. The first one concerns the PBX system selection in a manufacturing environment based on current traffic volume, user profile analysis, special feature requirements, and future growth. The major concern is the number of trunk lines required to support a specific probability of call blocking. The second case study analyzes the capacity of a health care information inquiry system with the goal of minimizing the time to service an incoming inquiry. The main concern is how many attendants should be used to handle various loads at different working periods (i.e. three working shifts, weekends, holidays, etc.).

CASE STUDY #1

A division of a manufacturing company is planning a relocation project which will move the entire division to a new facility fifty miles east of current facility. The division will operate their own data center which is linked to corporate data network via corporate PBX system. The existing voice communications service is provided by the corporate PBX network on a monthly rental basis. One of the major concerns is whether the division should continue to use corporate PBX network or operate their own PBX system.

Three factors need to be considered in order to make the "owning or renting" decision: traffic volume, current and future feature requirements, and the cost. The traffic statistics can be gathered from accounting records; feature requirements can be obtained from a user survey; and the cost is determined on current corporate billing invoices and future traffic volume and feature requirements.

Basic Traffic and Cost Statistics

There are eight departments in the division; they are grouped into two segments, called group X and group Y for this study. The voice traffic analysis is based on 178 phone extensions, some of them are dial-ups and some are data center operators. Following tables show the most recent six months (November 86 to April 87) cost information, including equipment cost and actual calling cost, and calling statistics.

Month	Group X ($)		Group Y($)		Equipment($)		Grand
	L.D.	Local	L. D.	Local	X	Y	total($)
Nov.	1,920	2,971	1,931	286	6,969	2,168	15,417
Dec.	1,061	4,085	3,562	255	6,730	2,168	18,262
Jan.	1,156	4,216	3,145	304	6,739	2,476	18,262
Feb.	1,112	3,843	2,766	310	7,118	2,573	17,722
Mar.	1,269	3,764	2,720	319	6,499	2,591	17,162
Apr.	1,763	5,524	1,225	303	6,994	3,346	21,072

Table 1 Six Months Cost Information

Month	#calls via PBX		#local calls		Total calls		Grand
	GroupX	GroupY	GroupX	GroupY	GroupX	GroupY	total
Nov.	2,296	1,672	2,969	1,045	5,265	2,717	7,892
Dec.	3,196	3,384	3,963	1,550	7,159	4,934	12,093
Jan.	2,993	2,785	3,415	1,359	6,408	4,144	10,552
Feb.	2,935	2,451	3,554	1,406	6,489	3,857	10,346
Mar.	2,709	2,446	3,337	1,223	6,046	3,669	9,715
Apr.	3,362	2,715	3,956	1,615	7,318	4,330	11,648
avg/ month	2,916	2,576	3,532	1,366	6,448	3,942	10,390

Table 2 Six Months Call Statistics

Month	Call $ per phone		# of Call per phone		Equipment $ per phone		Total $ per phone		Grand total
	X	Y	X	Y	X	Y	X	Y	
Nov.	39.1	43.5	50.6	53.2	67.0	42.5	106.1	85.9	192.0
Dec.	49.5	73.6	68.8	94.9	64.7	49.4	114.2	123.0	237.2
Jan.	51.7	65.1	61.6	78.2	64.8	46.7	116.5	111.8	228.3
Feb.	48.6	59.2	63.6	74.2	69.8	49.5	118.4	108.6	227.0
Mar.	49.8	58.4	59.9	70.6	65.6	49.8	115.4	108.3	223.7
Apr.	72.7	60.9	72.5	83.3	70.5	66.7	142.7	127.7	270.4
avg.	51.8	60.1	62.8	75.8	67.1	50.8	118.9	110.9	229.8

Table 3 Cost Per Individual Phone Statistics

Month	Long Dist X	Long Dist Y	Local call X	Local call Y	Dept. avg. X	Dept. avg. Y	Total average
Nov.	4.70	4.39	4.69	3.62	4.67	4.05	4.36
Dec.	4.82	3.94	4.76	3.10	4.73	3.66	4.20
Jan.	5.17	4.06	4.80	3.33	4.91	3.84	4.37
Feb.	5.15	4.15	4.77	3.51	4.87	4.05	4.45
Mar.	5.86	3.98	5.71	4.11	5.62	4.07	4.85
Apr.	7.11	3.95	7.33	3.31	6.89	3.81	5.35
avg.	5.47	4.08	5.34	3.50	5.28	3.91	4.60

Table 4 Individual Call Duration Statistics

From these tables, a phone user's profile can be interpreted as follows:

* A caller makes 67 calls a month
* Each phone costs $115 including equipment cost of $59 and phone call cost of $56
* Each phone call lasts 4.6 minutes and group X users tend to use the phone longer than group Y users (5.28 vs 3.91 minutes)
* Each call costs $0.84 on the average
* Each caller spends 47% on local calls and 53% on long distance calls, including 35% on corporate PBX and 18% on local telephone company's long distance services

Future Feature Requirements

A survey form was sent to each individual of the division in order to solicit information on desirable features and calling frequency and duration. The most desirable features, ranked by their voting popularity, are listed below.

* Call transfer
* Call forward in all case
* Conference call
* Call hold
* Busy override
* Data line for data/file transmission
* Call waiting indicator
* Automatic callback
* Call attendant
* Speaker phone
* Trunk queuing with auto routing
* Night answering

Tables 5 below lists the summary of call duration statistics.

	CALL DURATION (minutes) 1-3	3-15	15-60	60+	Total
# call-out	6,199.0	14,778.0	718.0	15.0	
# call-out/phone	40.5	96.6	4.69	0.1	
#call-out/phone/day	1.85	4.4	0.21	-	6.44
#call-in	6,289.0	14,656.0	976.0	21.0	
#call-in/phone	41.1	95.6	6.38	0.14	
#call-in/phone/day	1.87	4.35	0.29	-	6.51

Table 5 Forecasted Call Duration Data

The number of calls per phone per day is derived from the assumption that there are 22 working days in a month. The mean values for 1-3, 3-15, 15-60, and 60+ intervals are 2, 8, 45, and 61 minutes respectively. The weighted average call duration is 7.22 minutes. Therefore, an average phone user makes 6.44 outgoing calls per day and receives 6.51 incoming calls per day; or a person plans to use the phone 13 times a day with a weighted average call duration of 7.22 minutes.

PBX Capacity Analysis

A PBX's capacity is determined by the number of trunk lines it can support subject to a specific call blocking probability. A call blocking probability, as defined previously, is the chance that a phone usage is likely to be blocked due to an all-line-busy condition.

The common approach to analyze a PBX's capacity is to use the Erlang-B analytic model. An Erlang is the product of two variables: the number of calls per phone per unit of time and the call duration. Since April's statistics is 15% higher than the six-month average, it is used as a worst case scenario to determine the Erlang of current system. The forecasted data is used to estimate the Erlangs for future system. Both types of Erlangs are listed below.

	Call duration Current	Call duration Future	# call/min Current	# call/min Future	Erlangs Current	Erlangs Future
Long distance via corp. PBX	4.66	-	0.42	-	1.957	-
Long distance via phone co.	9.64	-	0.213	-	2.053	-
Local calls	7.33	-	0.58	-	4.251	-
Avg/total	7.21	7.72	1.213	1.44	8.742	10.468

Table 6 Current & Future Capacity Requirements

Based on Erlang-B model, the Erlangs and number of lines can determine the call blocking probability. The following table shows the call blocking probability based on 8.742 Erlang traffic.

# of trunks	Probability (%)
9	21.16
10	15.62
11	11.05
12	7.46
13	4.86
14	2.90
15	1.66
16	0.90
17	0.46
18	0.22
19	0.103
20	0.045
21	0.019
22	0.007
23	0.003
24	0.001
25	0.0004
26	0.0001
27	0

Table 7 Call Blocking Probability with 8.742 Erlangs

Modeled results indicate that for the traffic of 8.742 Erlangs; therefore, 27 trunk lines are required to support a non-blocking system. For retaining telephone industry's 0.01 standard, 16 trunk lines are necessary. Similarly, for the forecasted traffic of 10.468 Erlangs, 33 and 20 trunk lines are required to support a non-blocking and 0.01 blocking system, respectively.

Since the above modeled results are based on a combined total Erlang traffic, a different approach was used based on each individual Erlang traffic. Table 8 shows the modeled results.

Type of call	Long dist. via PBX	Long dist. phone co.	local calls
Erlangs	1.957	2.053	4.251
# of trunks	Prob. (%)	Prob. (%)	Prob. (%)
2	39.08	41.16	63.75
4	8.93	10.30	33.44
5	3.35	4.09	22.13
6	1.07	1.39	13.56
7	0.297	0.41	7.61
9	0.016	0.025	1.80
10	0.003	0.005	0.76
12	0.0001	0.0002	0.104
13	0	0	0.034
15			0.003
16			0.0008
17			0.0002
18			0

Table 8 Call Blocking Prob. from individual Erlang

Results from Table 8 indicate that in order to maintain a non-blocking probability system, a total of 44 trunk lines are required. For a 0.01 blocking system, 23 trunk lines are necessary.

Results from Tables 7 and 8 raise an interesting question. When all three types of calls are combined together, they require 27 trunk lines; when they are analyzed separately, the sum of trunk lines of each group is much larger than the combined requirement (44 vs 27). Why ? The combined situation implies that a group of trunk lines are available to be accessed regardless the type of calls made. The separate situation implies that a group of lines are dedicated only to a group of users. Therefore, the probability of blocking is higher in a separate situation than in a combined case. This brings up a decision making question and one must decide how to configure the trunk lines for achieving desirable call blocking probability. As discussed earlier, the trunk lines can be grouped together using either TIE, FX, WATS, or DOD/DID lines to realize the most cost effective routes.

Once the capacity requirements of a PBX are determined, the desirable features need to be considered for selecting a proper PBX system. Most telecommunications research reports can provide detail PBX description and functional comparisons.

CASE STUDY #2

A health care information services company provides a telephone inquiry services to the public. The current system consists of 16 lines and 10 attendants to answer inquiries, and 2 on-line CICS terminals for retrieving information. The inquiry system is operated as follows. If an incoming inquiry is not answered within 8 seconds, the call is placed in a pool answered by a first-come-first-service policy. There are two electronic priority indicators showing the next line to be answered. If a call is answered, the attendant may continue the conversation or retrieve information. The information retrieval is operated in two modes: manual and automatic. In a manual retrieval mode, an attendant goes to the hard copy looking for information; in an automatic mode, the attendant uses a terminal to access the CICS data base. Queuing occurs when an attendant confronts an all-terminals busy condition.

The desirable service goal is 10 minutes per inquiry including queuing and processing times. Since this goal is not 100% attainable, the management has ordered a PBX capacity analysis study to address following questions:

- What is the current system capacity ?
- If the system is to provide a 10 minute service per inquiry including a 3-minute initial waiting and 5-7 minute processing time, what is the necessary configuration to meet this goal ?
- How many CICS terminals are required for on-line inquiry processing in order to reduce queuing time on terminals ?
- How many attendants are required to handle the weekend and non-prime hours inquiries ?

Current Statistics

The inquiry system is monitored by a Code-A-Phone Sequence V monitor which operates in three modes: answer-only, answer-hold and idle. It provides detail telephone information on number of incoming and outgoing calls, number of abandon calls, and busy minutes. It also displays the number of calls answered by the monitor with a prerecorded announcement if the call is not answered after 8 seconds. The monitor also updates the status of queued calls. The measured statistics are:

	# inquiry/hr	Service time/inquiry
Weekday prime hours	75	6.74 min
Weekday non-prime hours	21	5.25 min
Weekend prime hours	8	5.25 min
Weekend non-prime hours	6	5.25 min

Table 9 Measured Inquiry System Statistics

The observed queuing statistics consist of three categories:

Avg. queue time for an available terminal	65 seconds
Avg. time difference between on-line and manual processing per inquiry	24 seconds
Avg. % of attendant waiting for a terminal	33%

The current system is capable of handling 60 inquiries an hour. The averaged prime shift hourly inquiry rate was 75 and approximately 25% of the incoming inquiries are placed in queues. This number is likely to increase as time evolves since the incoming rate is greater than the service rate.

Model Application

The telephone inquiry system is a classic multiple-server queuing system where callers enter a common queue if they are not answered immediately, and wait for the next available server (attendant). The input parameters to the model are: the average service time, number of available attendants (servers) to answer the inquiry, and the number of calls per minute (or hour). The model provides the following output information:

* Server utilization (how many minutes that an attendant is busy in an hour).
* Caller's queuing time.
* Number of callers in queue.
* Number of callers being serviced (answered).
* Total processing time per call including queuing and processing time.
* The probability that all servers (attendants) are busy.

The model is executed in iterative mode with varying number of attendants so that the results can be plotted for quick reference.

Modeled Result Analysis

As indicated earlier, the multi-server model can estimate the queuing and processing times corresponding to various load conditions in terms of number of inquiry per hour. Table 10 below shows these calculated values, in minutes, for the weekday prime shift hours with an average service time of 6.74 minutes.

# people	P = 10		P = 12		P = 14		P = 16	
Inquiry per hr	wait time	total time	wait time	total time	wait time	total time	wait time	total time
60	2.97	9.71	0.38	7.12	0.06	6.80		
61	3.56	10.31						
62	4.08	10.83						
63	4.82	11.56						
66	8.93	15.66						
69	24.43	31.14						
70	56.31	63.06						
72			1.51	8.25	0.26	6.99		
78			3.27	10.01				
84			9.61	16.35	0.89	7.63		
87			27.28	34.02				
88			70.66	77.39				
90					1.69	8.43	0.32	7.06
96					3.52	10.26		
102					10.28	17.02	1.00	7.74
107					30.65	37.39		
111							2.99	9.95
114							3.75	10.49
120							137.73	144.47

Table 10 Modeled System Capacities for Prime Hours

Data indicate that the current system can handle 60 inquiries an hour to meet the desirable goal: 3 minutes waiting and 7 minutes processing. To add 6 more attendants, the system capacity is increased to 111 inquiries an hour. Figure 1 summarizes the data in graphic form.

Since there are times that an attendant may have to wait for the terminal, the management needs to know how many terminals should be added in order to improve information retrieval time. The model assumes that no more than 1/2 of all attendants access the terminal at the same time; it then estimates the expected queuing times using the number of available terminals and number of simultaneous access, as shown in Table 11.

# simultaneous access	Number of terminals: 8	7	6	5	4	3	2
8	0	0	0	5.29	27.5	*	*
7	0	0	0	2.78	11.9	534.5	*
6	0	0	0	0	5.7	38.9	*
5	0	0	0	0	0	14.1	*
4	0	0	0	0	0	5.9	62.7
3	0	0	0	0	0	0	16.9
2	0	0	0	0	0	0	0

Table 11 Modeled Terminal Wait Time

Note that an asterisk indicates that the queuing time is extremely high. The observed terminal wait time is 65 seconds in a 10-attendants and 2-terminals configuration. The modeled terminal wait time of the similar configuration is 62.7 seconds, a very close approximation. Based on these estimates, a total of 5 terminals would reduce the average terminal wait time to only about 5 seconds. Figure 2 summarizes the results of terminal wait time analysis.

From Tables 10 and 11, the recommended system configuration consists of 16 attendants and 5 terminals, and the capacity is 148 inquiries per hour with an average wait time of 3.1 minutes and processing time of 5.25 minutes. Figure 3 plots the calculated system capacities versus the number of available attendants.

The weekday non-prime hours and weekend prime and non-prime hours have the monitored hourly rates of 21, 8, and 6 inquiries respectively, and the current number of attendant is 3. The terminal wait time in those cases should be close to zero. The model calculates the capacities for those periods in a similar manner as it did for the weekday prime hours, but with different service time value. Table 12 lists various system capacities corresponding to those periods.

# people	P = 1		P = 2		P = 3		P = 4	
Inq. per hr	Wait time	total time	Wait time	total time	wait time	total time	Wait time	Total time
6	0.49	5.74	0.061	5.31				
8	0.82	6.07	0.097	5.35				
11	1.51	6.76						
12	1.99	7.25	0.276	5.53	0.043	5.29		
18	8.57	13.82	0.97	6.22	0.187	5.44	0.03	5.28
21	28.43	33.67	1.76	7.01	0.28	5.53	0.07	5.32
22	87.39	92.63						
24			2.87	8.12				
30			10.83	16.08	1.39	6.64	0.16	5.41
33			43.4	48.65				
36					3.54	8.79		
42					13.34	18.59	1.19	6.44
45					81.10	86.36		
48							3.72	8.97
54							25.26	30.51

Table 12 Modeled Non-Prime & Weekends system capacity

Based on these results, two attendants should be sufficient to handle weekend and holiday load up to 14 inquiries an hour. Three attendants are required to support the non-prime weekday hours with a processing capacity of 24 inquiries an hour. Figure 4 shows the weekend and non-prime system capacities.

In summary, six attendants are required to handle the weekday prime hour inquiry demand. Two to three terminals should be added to reduce terminal queuing time. Two attendants can support the weekend and holiday inquiry demand and three attendants are necessary to handle weekday non-prime hour inquiry demand.

CONCLUDING REMARKS

We have shown that the same analytic models used for performance analysis and capacity planning in data processing and data communications can be applied to voice communications. In the PBX capacity selection case, a simple Erlang-B model, together with surveyed feature requirements and cost information, can provide very useful capacity planning guideline. In the telephone information inquiry system, the classic multi-server queuing model can be applied to determined the processing capacity based on a specific threshold on total processing time.

REFERENCES

JANC84 Janca, P.C., "The Computer-to-PBX Interface," Business Communications Review, March, 1984

MART72 Martin, James, "System Analysis for Data Transmission," Prentice-Hall, Englewood Cliff, New Jersey, 1972

OPDE85 Opderbeck, H., "The Integrated Voice/Data PBX: Architecture & Products," Technology Transfer Institute, 1985

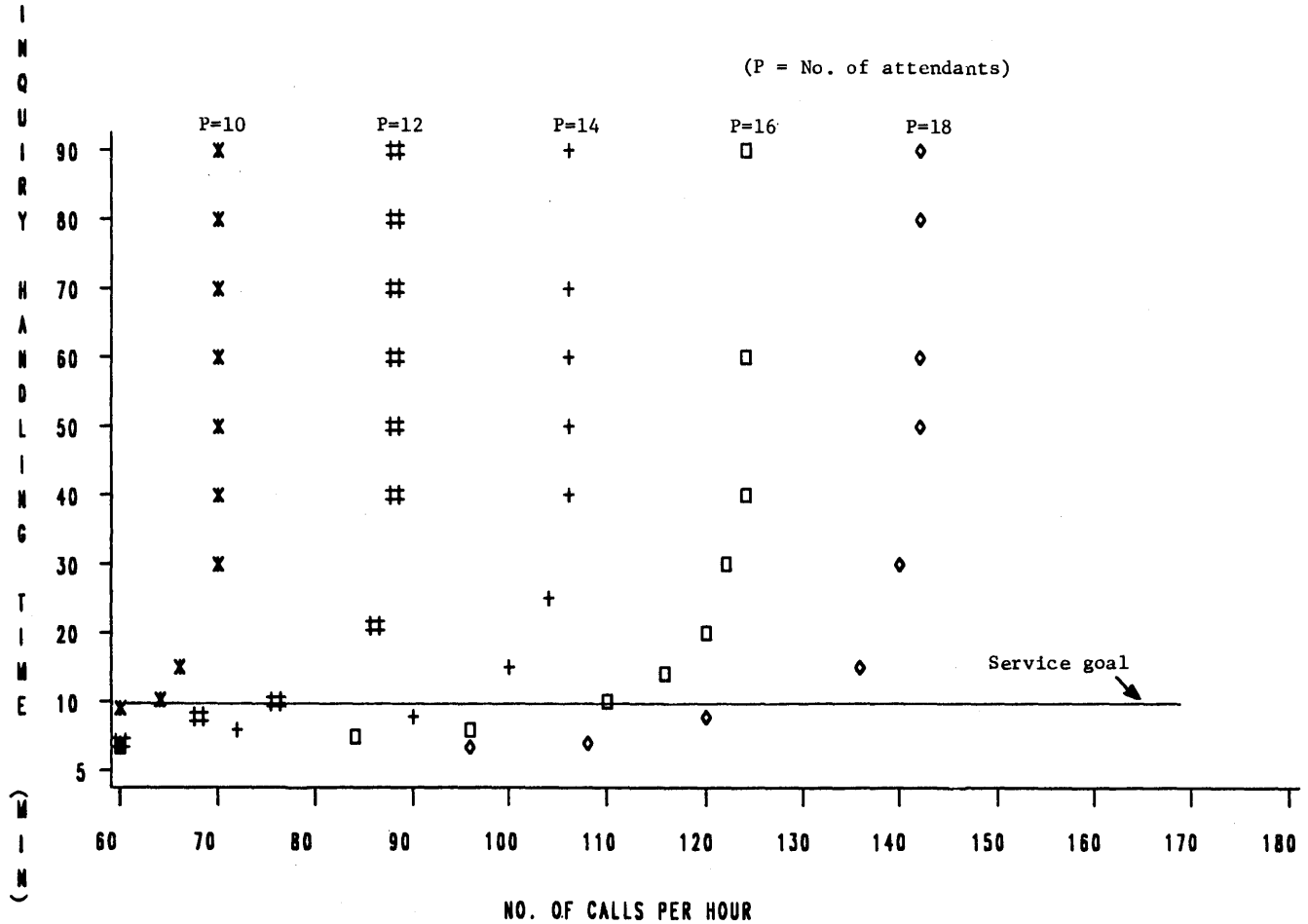


FIGURE 1 CURRENT PRIME-HOUR SYSTEM CAPACITY

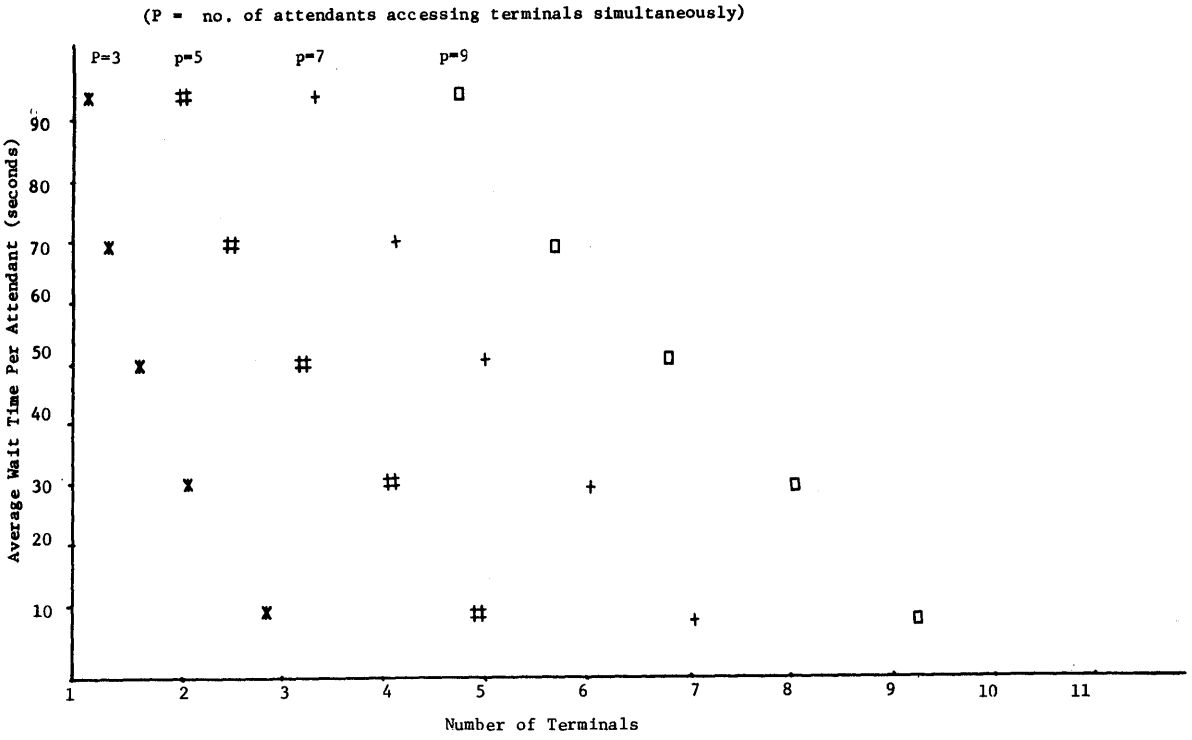


FIGURE 2 TERMINAL WAIT TIME ANALYSIS

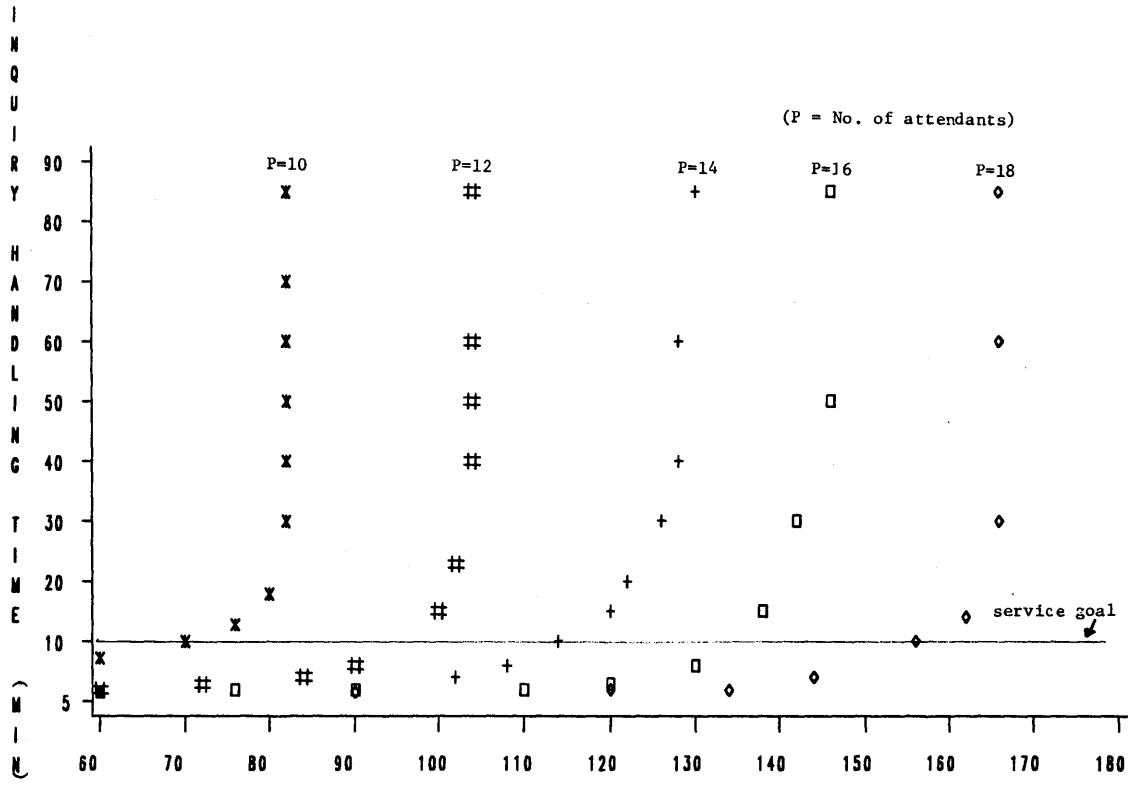


FIGURE 3 PROPOSED PRIME-HOUR SYSTEM CAPACITY

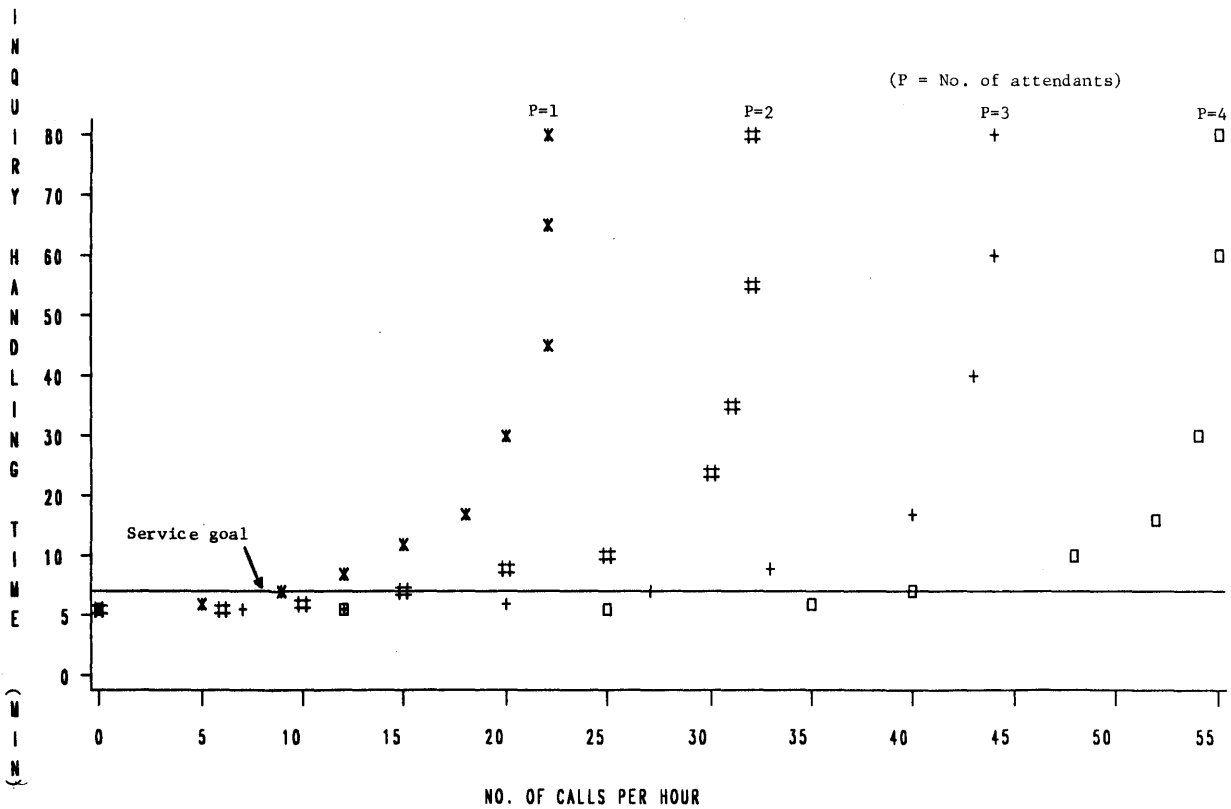


FIGURE 4 WEEKEND SYSTEM CAPACITY

CONCEPTUAL ISSUES IN MEASURING THE BEHAVIOR OF A DISTRIBUTED REAL–TIME PROCESS–CONTROL COMPUTER SYSTEM[1]

J. Dennis Lawrence
Lawrence Livermore National Laboratory

ABSTRACT

Measuring the performance, reliability and availability of a distributed real-time process-control fault-tolerant computer system presents a variety of interesting problems. A method is described by which this task can be organized efficiently, and by which appropriate measures can be provided for a variety of different user groups. This involves constructing a system structure graph that organizes all of the hardware and software of the computer system. Subgraphs are identified that provide information to the various users. A particularly important type of user is the equipment, or process, being controlled. Some of the problems that one can expect to encounter are discussed.

1. GENERAL PRINCIPLES OF COMPUTER SYSTEM BEHAVIOR

The behavior of any computer system can be understood quantitatively by providing certain measures of that behavior. This is true both for analyzing and explaining current behavior, and for predicting future behavior. However, behavior is only useful in the context of the tasks the system is expected to perform, and the objectives of the people who interact with the system. As a consequence of this, measures are context dependent, useful only with respect to the differing goals of the computer system users.

For example, consider measures of performance for a general-purpose timesharing system. A person sitting at a terminal is most interested in response time, while a person who runs batch jobs cares more about turn-around time. The system manager, intent on maximizing resource usage, will be quite interested in CPU utilization.

Now, suppose we have two people at interactive terminals. The first person is using the text editor and Fortran compiler to debug a large program, and the second person is directing time-consuming queries at a database management system (DBMS). The operating system (OS) stays up all day, but the DBMS goes down once an hour, and stays down for half an hour each time. How do our two users view this day?

To the first person, response time varies from fast to slow during the day, and seems to change a couple of times an hour. Still, the system availability is 100%, and reliability is perfect for the day. To the second person, however, response time is medium whenever he can get on, but availability is 50% and the mean time between failures is one hour.

You will notice that the goals of the three people in the first example are in conflict, since improving any one of the measures is likely to have a negative impact of the other two. Furthermore, performance, availability and reliability goals interact: an increase in reliability may result in improved availability and degraded performance. Or vice versa. As can be seen in the second example, even the perception of the system's behavior can vary widely among the users.

How can such goals and perceptions be reconciled? The answer to this question, in the setting of a distributed real-time process-control (RT–PC) computer system, is the subject of this paper.

We will be concerned with various aspects of three classes of behavior: performance, availability and reliability (frequently abbreviated to PAR). We begin by defining these terms, developing some general methods of measuring behavior, and providing an example. We will then go into each of these aspects in turn, considering measures appropriate to a distributed RT–PC computer system. In the process of doing this, we discover that there are important interactions that must be considered, and discuss methods of understanding these. Throughout, the emphasis is on obtaining practical answers to practical problems.

1.1. Definitions

The term *behavior* is sufficiently vague to refer to any aspect of a system that can be measured and is of interest to the observer. For this paper, however, we will limit the term to various aspects of performance, reliability and availability.

We will use the term *user* to include anyone or anything that interacts with the computer system. This includes persons that use the computer system as a tool to accomplish other work (frequently referred to as end users), and to persons (such as operators and systems programmers) who are directly concerned with the computer system itself. These will usually be referred to as *external users* and *internal users*; the distinction becomes important when we assign appropriate measures of PAR to different classes of users. A third type of user is the equipment being controlled by the RT–PC system.

Roughly speaking, performance refers to the speed with which the system responds to user demands; reliability, to its ability to operate without failure; and availability, to its ability to complete work for users, by satisfying resource requirements. While somewhat more precise verbal definitions can be given, it is necessary (to be useful) to specify these concepts in terms of mathematical equations that can be evaluated (ultimately) from observable measurements. For this paper, the following informal definitions are sufficient.

- *Performance* measures the efficiency with which a device or system satisfies service requests from (human or mechanical) users. A computer system can be understood as a collection of various hardware and software resources, connected together into a network. Activities consist of various tasks that are routed among the resources, requesting service. Each resource is limited in the number of tasks that can simultaneously receive service; other tasks may be queued, awaiting their turn for service. Performance measures the efficiency of all this activity.
- *Reliability* is the probability that a device or system will operate for a given period of time without failure under given operating conditions [1]. The reliability of a system is thus a measure of the success with which the system conforms to some authoritative specification of its behavior, and cannot be measured without such a specification [2].
- *Availability* is a measure of the ability of a device or system to complete a mission without excessive down time [1]. It is the expected fraction of time that the system is available to perform useful work.

See [3] for more precise definitions of reliability and availability.

[1]This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W–7405–Eng–48.

1.2. Classes of Measures

A first step in classifying measures (of all three types) results from separating external from internal measures. End users are generally most interested in external measures; systems management, in internal measures; and equipment, in both.

External, or *black–box*, measures are calculated by examining the behavior of the system from the outside, with no knowledge of the internal composition or configuration. These measures reflect how well the system is meeting current end user objectives, and thus should reflect current user satisfaction.

Internal, or *white–box*, measures are calculated by an examination of the structure of the system, and the behavior of the various subsystems and components into which the system can be divided. These measures have a variety of uses. Notice that each of these goals has application in each of the three aspects of PAR.

- to assist system management in reorganizing the computer system to better meet user objectives;
- to calculate how much additional stress the system can tolerate before user objectives can no longer be met;
- to predict and control system degradation as stress increases;
- to understand and predict the effects of changes in PAR on the equipment being controlled;
- to predict when new system resources will have to be acquired in order to improve behavior;
- to perform sensitivity analyses, so as to determine whether minor changes in system structure exist that can have major impact of system behavior; and
- to help find system bottlenecks and points of vulnerability, so effort can be directed towards critical areas.

You can see from this that external measures are primarily concerned with current behavior, while internal measures look more to the future, with the intent of decreasing uncertainty and improving management and operational decisions. External measures can be determined by actual measurement of the computer system, while internal measures combine these measurements with additional measurements internal to the system, estimates of future workloads, and anticipated future user requirements in order to predict and control future system behavior. This is generally done by constructing models of possible configurations, solving the models for various known or hypothetical workloads, analyzing the solutions to obtain insight into behavior, and using the insight to guide the actual evaluation and evolution of the system.

A brief example may indicate why we separate external from internal measures. Suppose we have a process-control system, which has two CPUs that control the equipment. Either CPU is capable of performing this control by itself, so if the primary CPU fails, the backup CPU takes over with no loss of control. Now, suppose CPU–1 fails. Repair takes some period of time, during which CPU–2 controls the equipment. When repair has been completed, CPU–1 becomes the new backup CPU.

Now, from the viewpoint of the external users, availability is 100% throughout this period of time. However, from the viewpoint of the internal users, only half of the potentially available system is actually there during the repair interval. The equipment continues to operate, but at increased vulnerability to failures. Both internal and external viewpoints are valid, and both are important. They cannot be combined; hence, we keep both external and internal measures when describing the system behavior.

1.3. Components, Systems and Views

A computer system can be considered to be a hierarchy of (sub)-systems and components. Here, a *component* is considered to be atomic – it is repaired or replaced as a unit, and we do not look inside it to measure behavior within. You can see that what one person thinks of as a component may be a complex system to someone else. In such conflicts, let us assume that agreement can be reached, so (in effect) a component is a piece of the system that nobody wishes to subdivide in order to consider PAR.

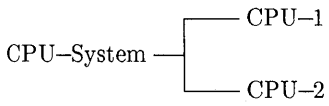
Components are combined with other components (and, possibly, with subsystems) to form subsystems. These in turn can be combined to form other subsystems, etc., until we arrive at the top of the hierarchy with a single system including everything. We permit components and subsystems to belong to more than one (sub)system; the resulting structure is a directed acyclic graph. We call this graph the *system structure graph* (SSG).

In practice, a component is either a hardware device or a software module. A hardware component would then be the smallest device that is removed as a unit for repair; this is meant to imply that resistors are not usually considered to be components (although we could, if we wanted to). A software component could be a single subroutine or a single program, depending on how deep into the system structure we wish to go. Keep in mind that our purpose is to obtain practical measures of PAR; reliability of a single resistor, or performance of a single line of code, is generally not helpful to the users of a computer system.

It would also be possible to add users (people and equipment) to the list of components, since their actions may well affect PAR. This idea is not pursued further here; we view people as demanding levels of service, and the system as attempting to provide those levels. If a user causes a system crash, we measure the crash, and assign fault to the component or subsystem that failed, not to the user.[2]

Now, suppose that there is some measure b of behavior that we wish to calculate for an SSG. Typically, certain measurements are carried out at some level within the graph. For example, if b measures reliability, we could begin by measuring reliability of individual components. These measurements are then combined, bottom to top, so as to obtain a measure that quantifies this behavior for the entire graph. Note that the evaluation of b at an internal node of the graph will ideally require information only from nodes below that node, but not nodes above it or in other branches. Unfortunately, this ideal is not always satisfied; behavior in remote portions of the SSG will frequently have an impact on the internal node for which measures are being calculated.

Different measures, of course, will require different algorithms. Recall the two-CPU example from the last section:



If our measure is external availability, A_E we have A_E(CPU–1) = 0.0, A_E(CPU–2) = 1.0, and A_E(CPU–system) = 1.0. However, if our measure is internal availability (defined as percentage of total capacity), we have A_I(CPU–1) = 0.0, A_I(CPU–2) = 1.0, but A_I(CPU–system) = 0.5.

An additional complication may occur if a measure can be determined at an internal node of the SSG, but not at any subordinate nodes. As an example, consider a communication system made up of several distinct links, each of which is

[2]The opposite point of view is quite legitimate, but is outside the scope of this paper.

tracked for reliability. It may happen that noise on one of the links causes end-to-end communication failure, but (for some reason) we cannot determine which link actually failed.

Typically, any given user is concerned with only a portion of the full SSG. If you make no plots, for example, then the reliability of the plotter is uninteresting. For this user, we can construct a subgraph of the SSG known as a *view*. The view contains only those portions of the SSG that are required for the actions desired by this particular user. As was pointed out in an earlier paragraph, calculation of measure b for this view will generally require reference to portions of the SSG that are not in the view. Views that correspond to portions of the equipment being controlled are, of course, particularly important.

When constructing subgraphs for internal users, it is possible to create *artificial* users to represent critical portions of the SSG that are not visible externally, and that do not directly involve the equipment being controlled. For example, we could define such a user to represent a critically important portion of the communication system, in order to track the performance and reliability of this resource over time.

There are a number of advantages to the technique outlined in this section.

- It accounts for legitimate variations among the various users. Since each user receives a separate set of measurements, it is possible for Management to objectively detect users that are receiving inadequate service, and to direct scarce resources to improving the situation.
- System Management can examine trends for different users, and project these trends into the future. In this way, it should be possible to predict degradations in PAR before they occur (in many cases), and to take remedial action in a timely fashion.
- By considering the equipment being controlled as a type of user, we can use the same technique to isolate portions of the system that are critical to the real time control system. In an RT–PC system, we cannot afford to lose track of these areas!
- The method should be efficient computationally, since there will be large areas of the SSG in common among different users. With proper organization and preparation, much of the data collection and the calculation of measures can be automated. Note, however, the importance of planning.
- The concept is very flexible. It is easy to add new external users, to redefine views as the needs of a user change, and to add new artificial users to isolate and study portions of the system that suddenly become critical.
- By maintaining historical data on the SSG, one can perform retrospective studies as well. This can be useful, for example, in understanding the background of a newly emerging problem area.

2. A TYPICAL EXAMPLE

An example may clarify the discussion in the last section. This example consists of a network of Digital Equipment (DEC)[3] VAX computers that manage a large number of DEC microcomputers (LSI–11's) that control individual pieces of equipment. In this system, each LSI–11 devotes itself to controlling its own portion of the overall process, providing status information to the VAX network, and accepting commands from the VAXs.

[3]DEC, DECnet, DECserver, MicroVAX, and VAX are trademarks of Digital Equipment Corp. Ethernet is a trademark of Xerox Corp. XYPLEX is a trademark of Xyplex, Inc.

The VAXs are used to coordinate the activities of the LSI–11's, subject to internally programmed criteria and commands from operators. There are a number of operators, each responsible for some aspect of the equipment that is being controlled. The VAXs provide detailed or summary information to operators, and accept commands from operators; these commands may, in turn, result in further commands from VAXs to many microcomputers, subject to the programmed constraints already mentioned.

The hardware for this example is sketched in Figure 1, a simplified version of an actual network at LLNL. A portion of the SSG is shown in Figure 2. Detailed information about the various devices can be found in references [4–6]. This description is intended merely as an example of a typical real time system [7], not as characteristic of all RT–PC's.

The actual network from which this example was derived currently consists of three dozen microcomputers; twelve controlling VAXs; eight Ethernet cables connected together by repeaters and (in some cases) fiber optic lines; over thirty direct-coupled terminals, many with touch panels attached; and numerous disks, printers and other peripherals. In addition, there are several dozen other computers on the same Ethernet (with a wide variety of uses) and over 400 terminals attached via terminal servers.

3. PERFORMANCE MEASURES

3.1. Introduction

Performance modeling of computer systems has received a considerable amount of attention over the past decade or so, and remains an area of active research [8]. Most of this work is directed at two common problems: general purpose time sharing systems and communication systems [9]. Almost all of the theoretical studies and computer modeling programs are based on queueing network theory – and the current study is no exception. For an alternative approach based on Petri Nets, see [10].

The general approach to modeling performance of a computer system is, at least in theory, straightforward and well understood [11]. The first step is to understand the structure and dynamics of the system; this results in a characterization of the workload and a queueing model of the system activity. Once the model has been developed, it must be solved for measures of interest, such as response time and utilization of the different components and subsystems. The general approach for a RT–PC system follows this model, but with some differences in emphasis, as discussed in the next subsection.

3.2. Comparing Real–Time Systems to Other Types of Computer Systems

Real-time process-control (RT–PC) systems differ, in a number of areas important to modeling, from time-sharing (TSS), and communication (CS) systems. In some cases, these differences lead to an easier problem to solve; in others, the reverse. Note that these differences are matters of degree, not of kind, and that many of the statements made below are generalizations. The existence of exceptions should be understood, even where not stated explicitly.

- System Usage.
  - Modeling emphasis for TSS is, of course, on interactive and batch use of a central computer complex. Jobs are submitted to the computer from remote terminals and/or batch stations at the whim of the end user.
  - Modeling emphasis for CS is on message traffic over a network. For this kind of problem, the computers on the network are generally of interest only as sources and sinks for messages. Message generation and routing are at the discretion of the end users.

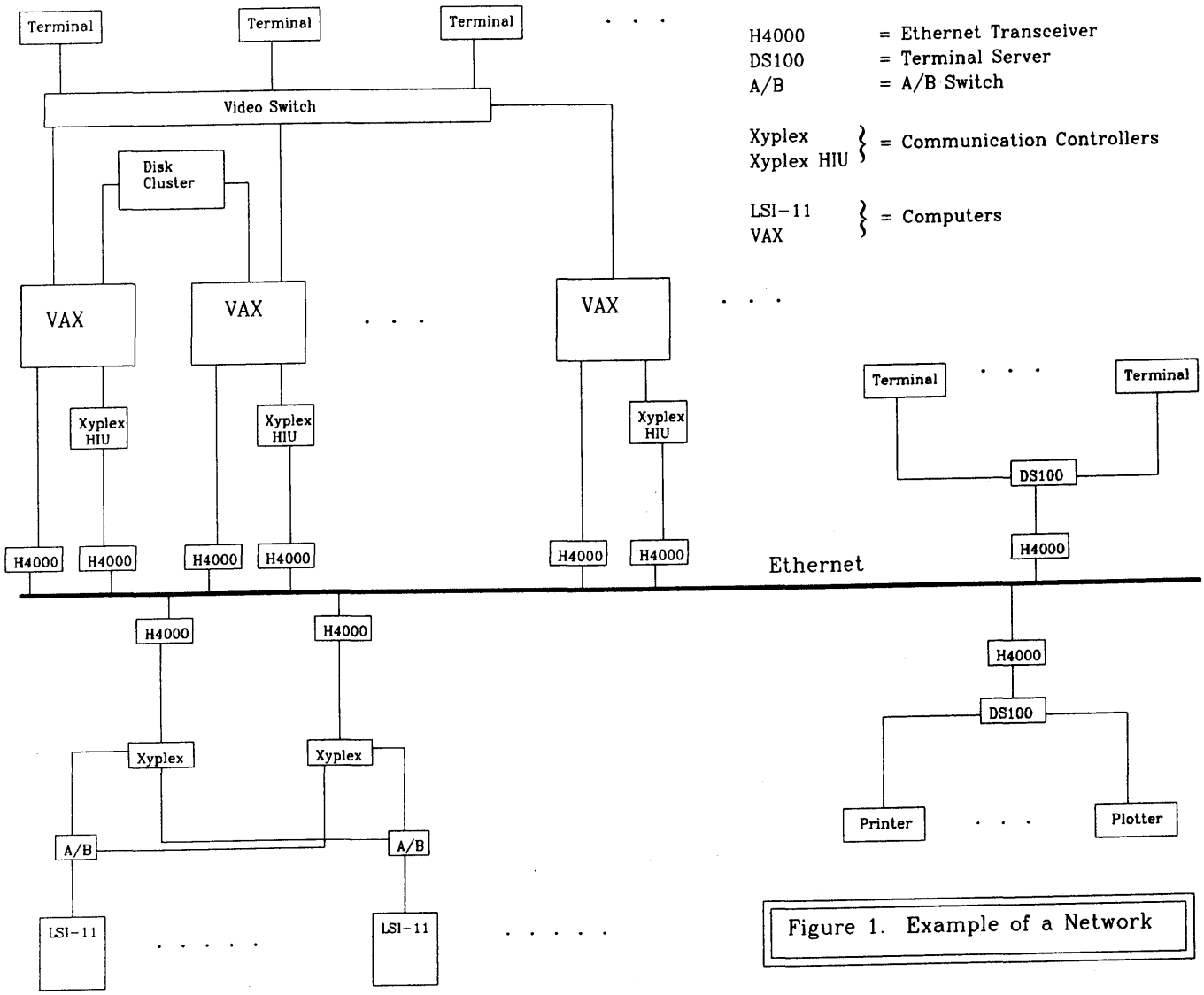


Figure 1. Example of a Network

– In a model of a RT–PC system, all aspects of the computer system are important. However, most of the actual programs run by the computer are fixed, and can be analyzed quite thoroughly.

System Structure.

– A standard TSS model is a single central CPU; I/O devices such as disks and tapes; and interaction devices such as terminals, card readers, and printers. This is sometimes referred to as the central server problem.

– A CS consists of a network connected to multiple CPUs and (possibly) terminals. The types of networks can range from local area networks, such as an Ethernet system, to very long range systems involving land lines and satellite communications. Arpanet is an example of the latter.

– A RT–PC problem of the type being considered here includes all of these aspects: multiple CPUs, possibly of different types; local area networks; analog and digital sensors and actuators; disks and (sometimes) tapes; terminals and other interactive devices, and (possibly) long range networks.

Time Span of Attention.

– Analysis of a TSS or CS system is generally aimed at long range averages. The model is designed to answer such questions as, "What is the average response time during peak hours of the day?" The unpredictable nature of the workload generally precludes detailed analysis, although this is sometimes of interest.

– Analysis of a RT–PC system is aimed much more at short range issues. The predictable nature of the workload, plus the necessity of keeping up with the equipment being controlled, make this both possible and necessary. A matter of particular concern is understanding the performance of the system during times when part of the system has failed, and during times of unusual stress from the underlying equipment.

Workload Characteristics.

– Workloads for TSS and CS problems tend to be both highly variable, and unpredictable in detail. This variability can be over a variety of time periods. For example, workloads tend to peak during certain periods of the day (say, 10 am to 4 pm), week (low weekend loads), month (end of the month processing), or year (tax time). Knowledge of such workloads is almost always statistical in nature.

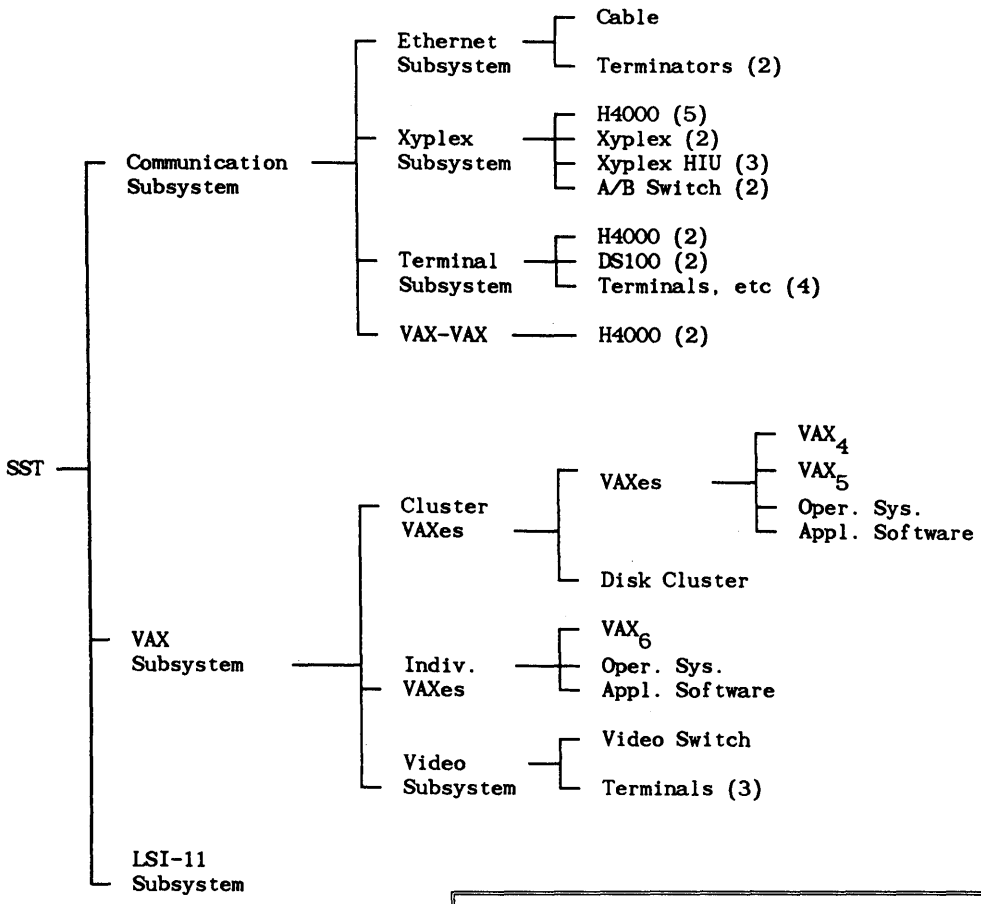


Figure 2. Example of a System Structure Graph

- Workloads for RT–PC systems frequently can be analyzed much more precisely. Any changes in workload will be caused by changes in the underlying equipment, by failures in some portion of the computer system, or by well-controlled changes to the computer hardware or software. Such changes can, therefore, frequently be predicted as precisely as necessary. In the extreme, one can look at the actual instructions of the various code modules.

- Type of Statistic.
  - Analyses of TSS and CS systems tend to be limited to means and variances of measured quantities.
  - Predictability is of prime importance in an RT–PC system. As a consequence, while means and (especially) variances are quite interesting, the analyst must be vitally concerned about worst-case behavior. The 90^{th} percentile may well be more useful than a mean.

3.3. Example

Let us examine some of the objects in Figure 1, in order to describe how performance might be measured.

- Ethernet. The primary measures of interest are: the number of collisions per unit of time, wait times (due to the detection of collisions and the binary exponential backoff algorithm used to schedule retransmission), and channel utilization [12–13]. Commercially available products exist for measuring traffic and collisions. Results can be used to create input to a modeling program such as PLANS [14–15], in order to estimate future loads.
- H4000 transceiver and A/B Switches. These devices are totally transparent to messages, and can therefore be completely ignored when analyzing performance. Note that this is not true when analyzing reliability.
- DS100 terminal server. This device has one connection to an H4000 and up to eight connections to serial devices such as terminals. Primary functions are: message buffering, speed adjustments, detecting and resolving Ethernet collisions, and format changes between streams of characters (to/from the serial devices) and Ethernet packets. If modeling is necessary at all, it is quite straightforward, and quite conventional.
- VAX. There is nothing conceptually unusual about building a model of a VAX, other than to say that the factors discussed in the last subsection must be considered. (This is not meant to imply that it is trivial.)

Once the performance of the individual hardware and software components is understood, we can use the SSG to construct more global models. Typically, a model can be developed at each internal node that represents the activity carried out at that node. Models developed for subordinate nodes can be included as "subroutines." The process continues until everything in the graph is included. At this point, the performance of the various views can be evaluated for the various users.

All this appears to involve quite a bit of work. Many models must be constructed, and various sorts of workloads analyzed. However, the models tend to be small, and the workloads would have to be analyzed in any case. As with many other problem areas (for example, writing computer programs), careful planning, formal design, and a subdivision of the problem into a hierarchy of sub-problems can lead to an elegant, efficient and manageable solution.

4. RELIABILITY MEASURES

4.1. Introduction

The difference between reliability and availability, and the importance of each, can be illustrated by an example. The drum of a cement mixer must be kept in rotation, or the cement mixture hardens, and a new vehicle must be purchased. As a consequence, reliability is an important consideration in purchasing decisions for cement mixers.

Now, suppose the vehicle is connected to Headquarters via a microwave radio. As the truck moves about the city, radio blind-spots are occasionally encountered; during these, transmission fades for a few seconds. You can see that this communication system has low reliability (in the order of minutes), but high availability, since dead times usually don't last long. If the radio is used only to give routing instructions to the driver, this may well be a cost effective means of communication, and would be preferred to one with high reliability (months) that cannot be fixed, so must be discarded upon failure.

But, suppose the rotation of the drum is controlled by the radio link. No signal, no rotation. For actually driving about the city, this is probably ok, since inertia will keep the drum going during the brief dead spots. But, what happens if the truck stops for a traffic signal in a blind spot? A more reliable communication system may be preferred!

A RT–PC computer system has attributes of both aspects of this example. The portion of the system that directly controls the equipment must generally have very high reliability and availability. However, if a sufficient number of terminals are available to the operators, no one terminal will have such stringent requirements. A database system that collects status information at periodic intervals could require high availability (so that the probability is high of being available when such an interval occurs), but not such high reliability (since the status information is not collected continually).

4.2. Problems in Determining Reliability and Availability

A number of difficulties arise in attempting to measure reliability and availability of any computer system, including RT–PC systems. Once again, the differences between TSS and CS systems on the one hand, and RT–PC systems on the other, are more a matter of emphasis than of kind. Standard methods of reliability block diagrams and fault tree analysis are useful [16]. See [17–18] for more detail on computer-related reliability issues. The problems listed here are representative.

- Transient failures are a significant problem. Longbottom [17] points out that about half of the failures in a computer system are transient. It is frequently difficult to isolate the cause (since all components will show correct behavior when tested); however, the system has failed, and this must be accounted for somewhere.

- It can be very difficult to trace the cause of some failures. A series of apparently unrelated errors may eventually (after days or weeks of effort) be traced to a single fault. A particularly difficult aspect of this involves interactions among hardware, software, operations personnel, and the immediate environment, since faults in any one of these areas can sometimes show up as errors in any other area.

- The statistical theory of reliability depends on a number of assumptions, some of which are not satisfied in computer systems. The effect of such violations on accurately calculating reliability and availability measures can be obscure.

  - All failures are assumed to be statistically independent. In fact, however, computer failures tend to bunch. That is, after one computer failure, there is a significantly increased probability of additional failures for a short period of time.

  - It is assumed that the operating times from all instances of a component can be combined in order to calculate total hours of operation and total number of failures.

  - It is assumed that operational hours can be measured with reasonable accuracy. For purely electronic components, this presents no particular problems. However, there are difficulties in defining "operational" for mixed electronic-mechanical components, such as tape drives and printers, and more severe difficulties for software. Is a printer operational when turned on, but not printing? Is a program operational when not being run?

  - The action of repairing a component is assumed to have no effect on future failure rates. That is, a repaired component can be considered new.

  - Finally, sufficient failures of each type of component are assumed to occur to permit a failure distribution to be calculated, or (at least) to determine if an exponential distribution is a reasonable hypothesis for a distribution function.

- It is not always obvious how to include environmental failures caused by events outside the computer system, such as electric power failure, or interference by unrelated computers that happen to share the network.

- Faults may be detected by many components within the man-machine system. For example: a computer operator detects an operating system crash, the operating system detects an error in a disk read command, an application program module notices that a value from a sensor is impossible, a communication module finds a parity error in a message, and so on. Fault detection is a distributed activity.

A common method of improving the reliability of a RT–PC system is to use fault tolerant techniques throughout the system [19]. The intent of such techniques is to improve the external reliability of the system, by removing as many single points of failure as possible. This, of course, adds considerably to the overall complexity of the system, and to its analysis. As a result of this, computer help is required in order to construct and analyze a reliability model [20]. One promising technique, which can be used to analyze and understand hardware, software, operational and environmental reliability is that of fault tree analysis [16, 21–22].

In a conventional computer or communication system which makes no attempt to tolerate faults, the impact of reliability on performance is pretty obvious. Once fault tolerance becomes an important aspect of the system, the problem becomes more complex. It is in the very nature of a fault tolerant system to hide failures from end users, although degraded performance may be permitted. As a result, the performance model should be augmented to account for failures of different devices. An example of this is given in [23].

The system structure graph, and its various user-dependent subgraphs, is a useful tool in organizing the collection and analysis of failure data. The occurrence of a fault can be attached to any node of the graph, not just the leaves. This is useful in those cases where an error is seen in some subsystem of the RT–PC system, but cannot be assigned to any particular component. For example, suppose a message must be sent from a control computer to an actuator attached to one of the embedded microcomputers in Figure 1. It doesn't arrive on time, due to collisions on the Ethernet; this is detected by the microcomputer, and an appropriate message is sent back to the controller. We have a fault, but no one component has failed. The *subsystem* has failed, and that's where the fault should be attached for reliability measurement.

In many cases, there will be too few instances of failure of a particular type of component to permit accurate prediction of component failure rates. However, it may well happen that enough failures occur in some subsystem that includes the component (as well as many more) to permit determination of a failure rate distribution for the subsystem. In this case, we can measure the MTBF for the subsystem, but cannot predict why the different failures will occur. Here, too, the SSG is a useful organizing tool.

4.3. Example

Many examples of reliability measurements have been given above; only a few are added here. If we examine Figure 1 in order to measure reliability and availability, we will notice a number of differences from Subsection 3.3.

- Ethernet and other cables. Possible sources for reliability problems include: broken cable, short circuits in devices connected to a cable, and radio frequency noise in the environment. (There are others, of course.) The first two are hard failures, while the last may well be transient. A broken cable will prevent message traffic, while the short circuit could show up as spurious messages. Even in a simple cable, a variety of problems can occur, with a variety of symptoms.

- H4000, A/B switches and DS100. These devices may fail, so a MTBF exists, and can be measured if enough failures occur.

- Application Software. As we all know, application programs occasionally fail. Depending on the nature of the failure, and on the recovery method being used, the failure may have little noticeable effect, the entire system may crash, or anything in between.

- Operators. There are two broad categories of operator failure: (1) the procedures which the operator is expected to follow are correct, but the operator does something else (for any of a number of good or bad reasons); or, (2) the procedures are incorrect, and the operator may or may not follow them. A particularly interesting problem here is the case where the operational procedures are wrong, the operator detects this fact, the operator carries out a correct procedure, and then fails to notify anyone of the error.

- Environment. The variety of ways that the environment can interact with our RT–PC system, coupled with the infrequency of most of them, can lead one to neglect this aspect of failure analysis. In some cases (such as an explosion that levels the plant), it is perfectly proper to ignore the environment. However, in many other cases, an environmental analysis should be included. This is particularly true for power failure or surges, broken pipes, fire, and so on.

5. CONCLUSIONS

We have presented a general technique for analyzing the performance and reliability of a real-time process-control computer system, and discussed some of the conceptual problems involved in analyzing such systems. The primary ideas expressed in the paper are:

- There exist a wide variety of (human, equipment and artificial) users, with very different requirements. An analysis of a RT–PC system must be prepared to recognize each of them.

- Reliability, availability and performance of a RT–PC system must be considered jointly.

- The system structure graph is a useful device for organizing the information and calculations necessary to measuring the performance, reliability and availability of a RT–PC system.

REFERENCES

1. Smith, D. J. *Reliability Engineering*, Pitman (1972).

2. Kohler, W. H., "A survey of techniques for synchronization and recovery in decentralized computer systems", *ACM Computing Surveys* **13**, 2, 149–184 (June 1981).

3. Siewiorek, D. P., and R. S. Swarz, *The Theory and Practice of Reliable System Design*, Digital Press (1982).

4. *VAX Hardware Handbook*, Digital Equipment Corp. (1985).

5. *Microcomputers and Memories*, Digital Equipment Corp. (1982).

6. *The Xyplex System User's Guide*, Version 2.0, Xyplex, Inc. (1986).

7. Schoeffler, J. D., "Distributed computer systems for industrial control", *Computer* **17**, 2, 11–18 (February 1984).

8. Sauer, C. H., and K. M. Chandy, *Computer Systems Performance Modeling*, Prentice Hall (1981).

9. Sauer, C. H., and E. A. MacNair, *Simulation of Computer Communication Systems*, Prentice Hall (1983).

10. Marsan, M. A., G. Conte, and G. Balbo, "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems", *ACM Trans. Computer Systems* **2**, 2, 93–122 (May 1984).

11. Ferrari, D., G.Serazzi, and A.Zeigner, *Measurement and Tuning of Computer Systems*, Prentice Hall (1983).

12. *Introduction to Local Area Networks*, Digital Press (1982).

13. Shoch, J. F., and J. A. Hupp, "Measured performance on an Ethernet local network", *Comm. ACM* **23**, 12, 711–721 (December 1980).

14. Nishida, T., et al., "PLANS: Modelling and simulation system for LAN", in D. Potier (ed), *Modelling Techniques and Tools for Performance Analysis*, Elsevier Science Pub., 235–254 (1985).

15. Nishida, T., *PLANS User Manual for VMS User*, Dept. of Information and Computer Sciences, Faculty of Engineering Science, Osaka University, Toyonaka, Osaka 560, Japan (1986).

16. Pages, A., and M. Gondran, *System Reliability Evaluation and Prediction in Engineering*, Springer-Verlag (1986).

17. Longbottom, R., *Computer System Reliability*, Wiley (1980).

18. Siewiorek, D. P., and R. S. Swarz, *The Theory and Practice of Reliable System Design*, Digital Press (1982).

19. Wensley, J. H., "Fault tolerance in a local area network used for industrial control", *Proc. Real–Time Systems Symposium*, Dec. 6–8, 1983, IEEE Computer Society, 113–118 (1983).

20. Stiffler, J. J., "Computer-aided reliability estimation", in *Fault Tolerant Computing Theory and Techniques*, D. K. Pradhan (ed), Prentice-Hall, 633–656 (1986).

21. Leveson, N. G., and P. R. Harvey, "Analyzing Software Safety", *IEEE Trans. Software Eng.*, **SE–9**, 5, 569–579 (September 1983).

22. Hecht, H., and M. Hecht, "Fault-tolerant software", in *Fault Tolerant Computing Theory and Techniques*, D. K. Pradhan (ed), Prentice-Hall, 658–695 (1986).

23. Fernandes, V., J. C. Brown, D. Neuse, and R. Velpuri, "Some performance models of distributed systems", *Proc. of the CMG XV Inter. Conf.*, Computer Measurement Group (1984).

COMPREHENSIVE PROCESS FOR PROJECTING COMPUTER RESOURCES

John Wandzilak & Ray Murphy
Empire Blue Cross Blue Shield

ABSTRACT

The process used to predict computer resource requirements at Empire Blue Cross Blue Shield has evolved over the last eight years. It has proven to be a highly reliable and accurate methodology that drives our annually updated five year equipment plan. This paper offers in essence a "how to" approach to the computer resource planning process. Although the process was created in a health insurance corporation environment, the fundamental steps are applicable to all businesses. Figure 1 illustrates the necessary functions to support the process. Basically the process is 1) the translation of business indicators into related data processing workload demands, 2) the conversion of workload demands into alternative computing equipment configurations to meet service levels, 3) management approval of an equipment configuration based upon performance and financial considerations. The body of this paper follows the flow of Fig. 1.

BUSINESS FORECASTS

The first step is to develop a business forecast.

1. Generate business trends from a baseline of historical information for the past five years. The data should contain the basic profiles of marketing, operational and financial statistics. This data base will provide a ramp for launching future projections enabling the workload analyst to fully understand the inter-relationships of the business functions.

2. Some corporations have five year business plans indicating sales by product and area, production schedules, worker populations, etc. Such plans facilitate the business forecast. Unfortunately, many companies do not have long range quantifiable data. Quite often business plans are available only for the current budget year, but can form the bridge to an estimate for outer years.

3. In the absence of hard forecasting data, trends can be generated based upon the historical database. Industry trends, economic conditions, unquantified but stated corporate goals, budgeted projects and products that influence the shape of the business projections and their relationships must be understood.

4. Develop a model of the business based on the data collected. The model should be simple with only a few of the major business indicators. At Empire key business drivers are contract volumes by type, members per contract, and related claims volume. An analysis of this data is required to substantiate basic business relationships (e.g. claims volume do not necessarily linearly track contract size). It is vital to explore the effect of one business parameter on another.

5. Establish basic business assumptions, draft a numerical forecast, and then meet with the corporate business experts in marketing, finance and operating areas to validate the data.

6. Prepare a final draft of the business forecast.

WORKLOAD PROJECTIONS

The workload projection step is very similar to that of the business forecast.

1. Develop a baseline of data processing workload information. A five year historical file of data processing utilization metrics should contain:

   a) On-Line transactions

   By type, volume (prime and other shifts), response time

We use a customized software package to capture on-line transaction volumes by type, volume by response time, by hour of day. (See Fig. 2)

   b) CPU hours
   By application type, test and production, on-line and batch, by shift, by machine.

The CPU hours are collected by an accounting package which processes SMF daily logs. A key factor to the success of our workload forecasting process is the establishment and on-going control of appropriate job codes for each application, by production batch, production on-line, test batch, test on-line and support categories.

2. The creation of the resource utilization data base is followed by the difficult task of relating computing metrics to business drivers. Relationships are derived from the merging of the business data base with that of computer resources used for production functions:

   -business drivers to CPU hours, to on-line CPU hours, and to the number of on-line transactions.
   -on-line transactions to on-line CPU hours.

Fig. 3 shows how a particular type of claims volume e.g. medical claims relates to the CPU used in a batch mode by the medical claims system. Fig. 4 displays on-line transaction volume vs CPU time. These types of relationships need to be determined for every batch and on-line production system.

Relationships also need to be determined for testing. Although testing usage for a development project level varies greatly with the point of the project life cycle, maintenance efforts take on a more fixed relationship. Year after year we have concluded that the total envelope of testing utilization is related to population of application systems staff with adjustment for major high priority projects.

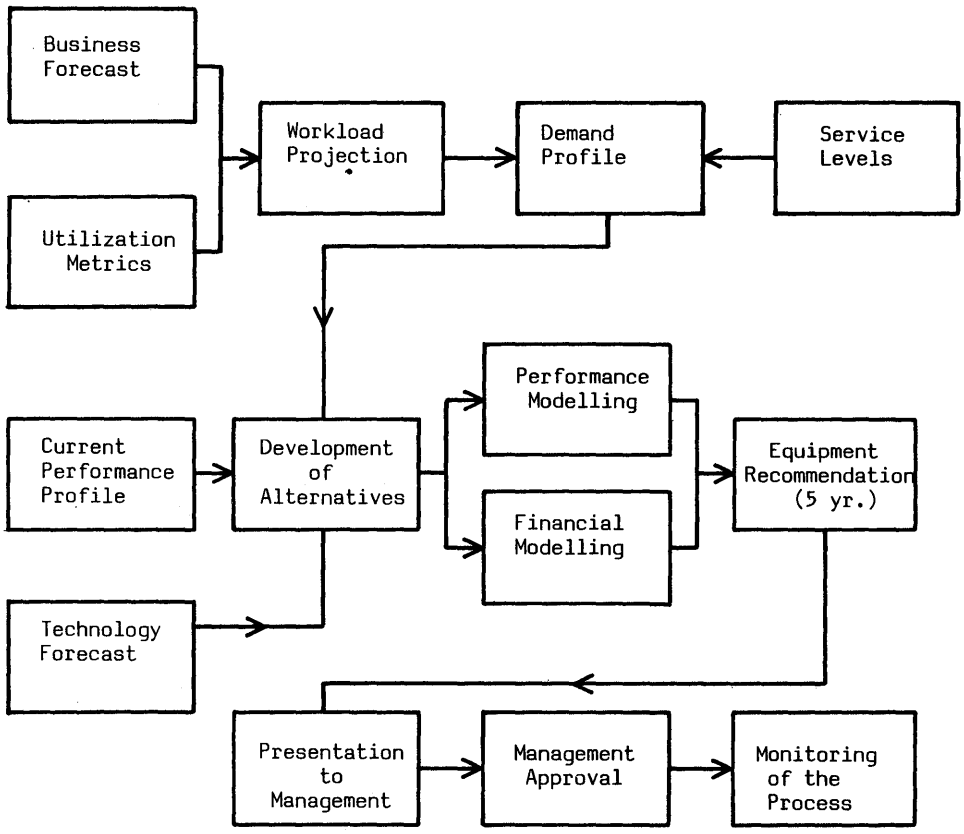


Figure 1.

TELEPROCESSING REPORT
APPLICATION REPORT **DAYTIME** 05/11/87 TO 05/14/87

VOLUME BY APPLICATION

RESPONSE TIME (SEC)	8-9AM	9-10	10-11	11-12	12-1	1-2	2-3	3-4PM	TOTAL 8AM-4PM
				MCS-MISC					
0- 2.9	671	1,443	1,173	1,220	1,103	860	1,375	1,292	9,137
3- 3.9	2	2	6	2	6	4	5	9	36
4- 5.9	0	1	6	5	3	3	1	3	22
6+	1	5	4	6	3	3	0	6	28
TOTAL	674	1,451	1,189	1,233	1,115	870	1,381	1,310	9,223
				MCS-MOFR					
0- 2.9	4,884	8,702	8,833	8,493	6,687	6,413	8,805	8,577	60,394
3- 3.9	90	236	449	386	146	110	297	427	2,141
4- 5.9	28	174	321	171	49	41	91	340	1,215
6+	8	78	161	48	27	9	16	205	552
TOTAL	5,010	9,190	9,764	9,098	6,909	6,573	9,209	8,549	64,302

Figure 2.

3. We now can proceed with developing a future demand profile. Each on-line transaction volume is sized dependent upon the related business driver and adjusted for any anticipated increased functionality. Validation is accomplished through separate meetings with user area representatives and application systems project leaders by the workload analyst. Similarly, CPU hours are projected by sizing the impact of business drivers as well as system functionality. Changes to the computing environment, such as upgrades in CPU's which require the normalization of CPU hours, and new software (SNA moved some network functions and related CPU hours from the CPU to the communications controllers), must also be weighed. Test demands are estimated by reviewing near term and long term project schedules and staffing, and using historical data as benchmarks.

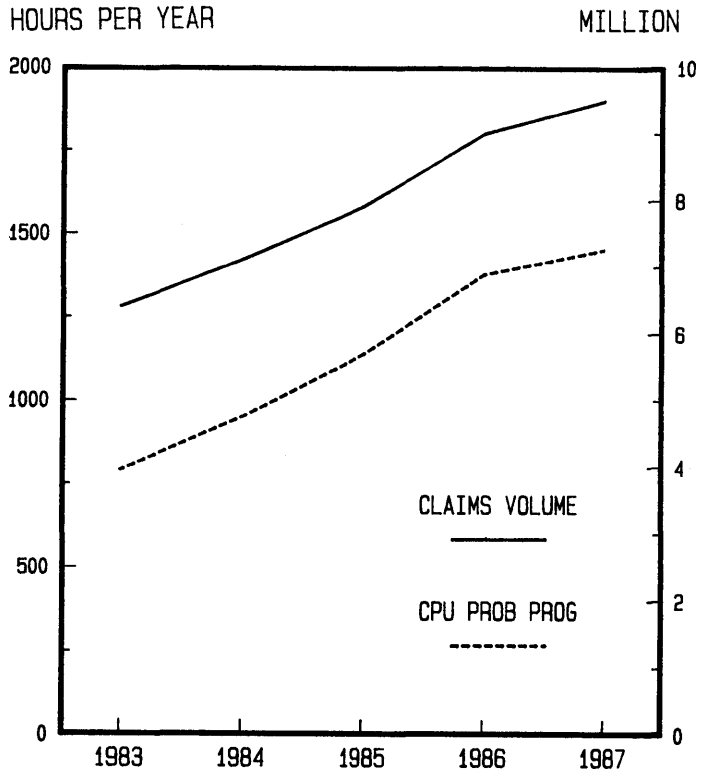
4. The workload forecast is then created by summarizing all of the details by category:

   - on-line production
   - batch production
   - on-line testing
   - batch testing
   - support e.g. technical services functions, quality assurance etc.

SERVICE LEVELS

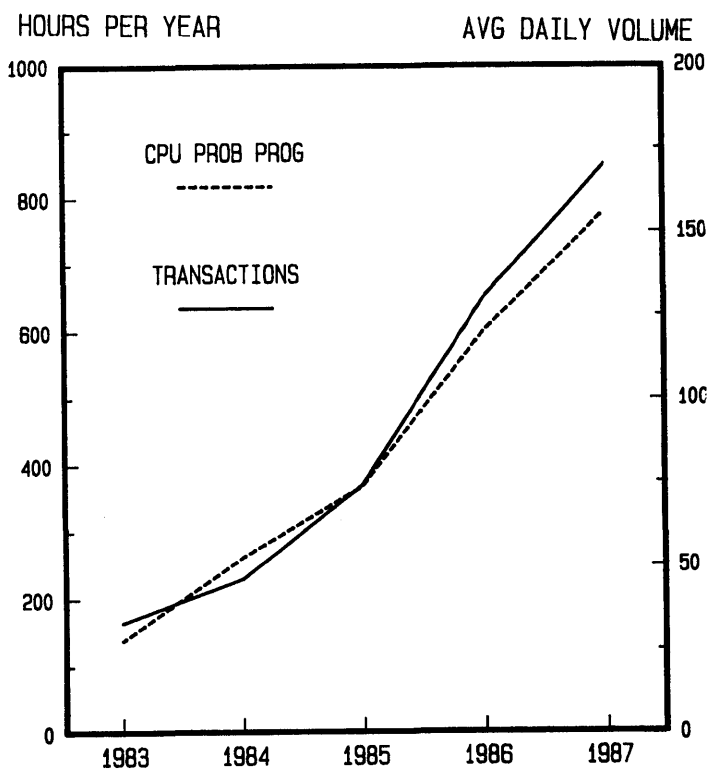
The projection of workloads based on accurate business and utilization data establishes a demand profile only at a particular service level. If on-line transaction levels are needed by the user community at a faster response time than currently experienced, then compliance with higher service level will change the demand profile. It is important to understand the details of all service changes. For instance, an increase in an on-line application's volume, functionality and/or service level may not have any impact if it occurs in the early evening. However, a small change during the prime shift may create significant service problems. A formal service level agreement (SLA) program is in effect with all of our major users. Service level performance vs goals are continuously monitored and the resulting data is used to form a basis for user/data processing performance reviews.

New service level requirements are incorporated into the overall demand profile.



MEDICALS CLAIMS VS MCS BATCH CPU

Figure 3.



MCS ON-LINE

Figure 4.

EQUIPMENT PLANNING PROCESS

Four different elements feed the development of alternatives for addressing the demand. The demand profile, as previously discussed provides a basis for determining future requirements. However, service levels, which are formally defined by SLA's, also influence the alternatives. Frequently, the service levels which are in place in current SLA's may not be adequate for the five year planning period. Improvements in service may be desirable (See Fig. 5) and, if so, must be factored into the demand profile at this point. (Any commitment to improve service over the planning period will by necessity increase the demand requirements). It is essential that service commitments which form the basis of the plan be agreed to by responsible senior management early in this process. The current workload must be broken down into classifications, and growth projected separately in each classification, to determine total requirements for the planning period (See Fig. 6). Finally, a technology forecast is undertaken. Each major processor vendor is contacted and discussions regarding present and future product plans are held. These directions are formally documented in the plan and become key in developing alternate processor schedules.

DEVELOPMENT OF ALTERNATIVES

At this point, demand projections and service level requirements are translated into peak period MIP requirements for each year of the planning period. The translation to peak requirements is key. While measurements and projections of demand are often made on a shift or 24 hour basis, the equipment must be configured to handle the demand of sustained 15 minute peaks in the 10:00 a.m. - 12:00 noon and 2:00 p.m. - 4:00 p.m. periods.

After peak MIP requirements are developed, (See Fig. 7), alternative processor strategies to meet this demand are generated. The technology forecast must be used to insure appropriate technological direction and proper use of vendor offerings in each strategy. Some basic criteria (cost, vendor viability, match with demand, etc.) are applied to reduce the number of alternatives to three which will be further evaluated.

MIPS REQUIREMENTS FOR IMPROVED SERVICE

(Alternative A, 1986-1991)

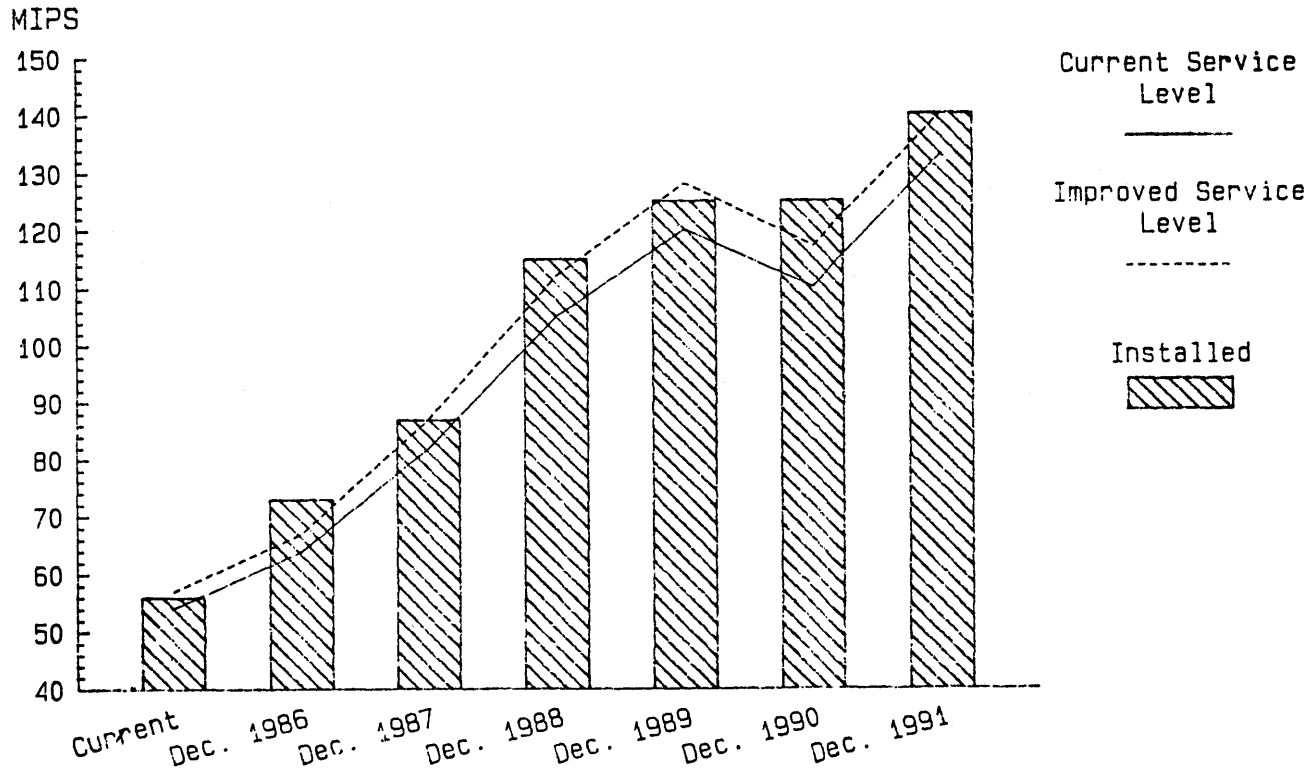


Figure 5.

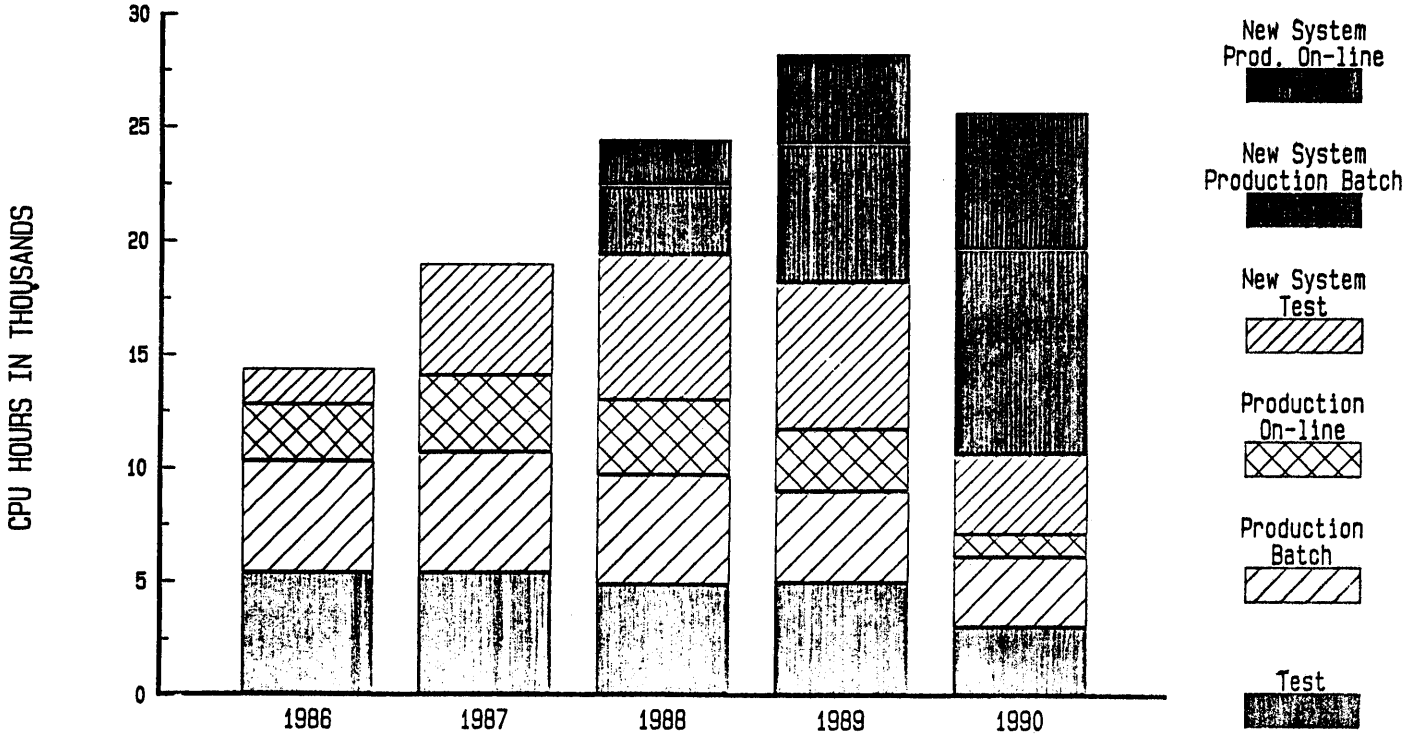
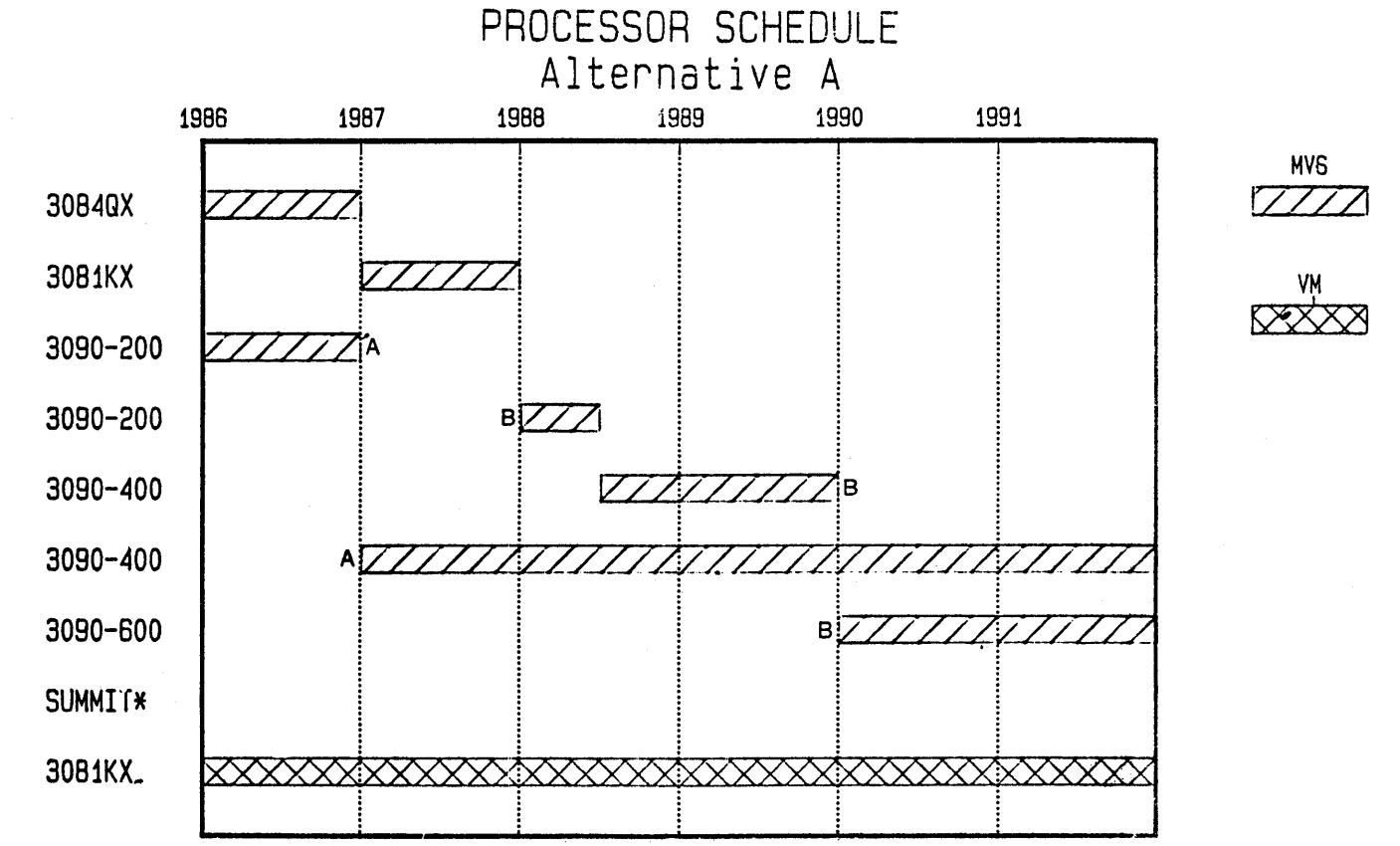


Figure 6.



*SUMMIT is an IBM unannounced 90-100 MIPS processor; forecast in 12/91.

Figure 7.

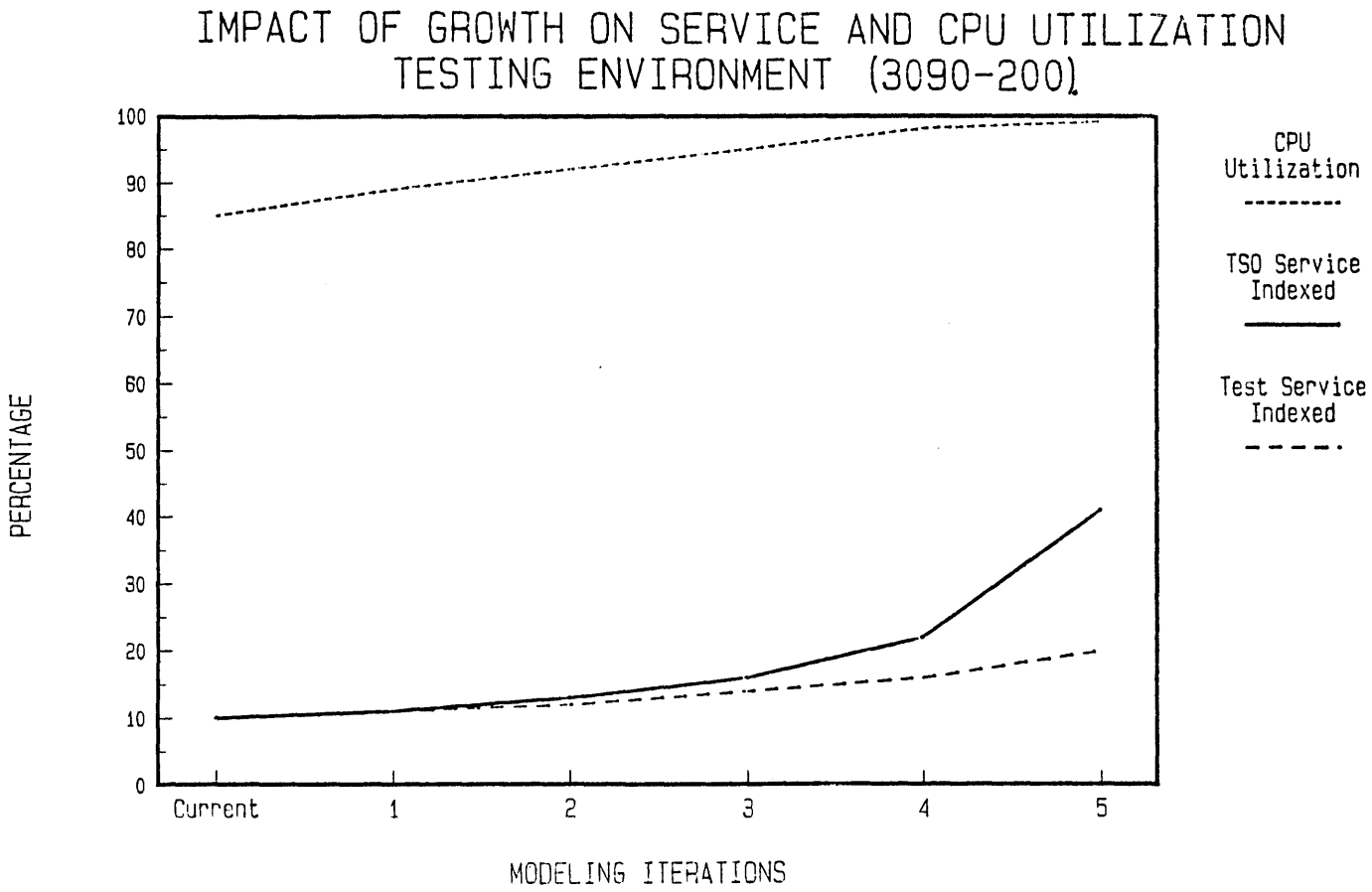


Figure 8.

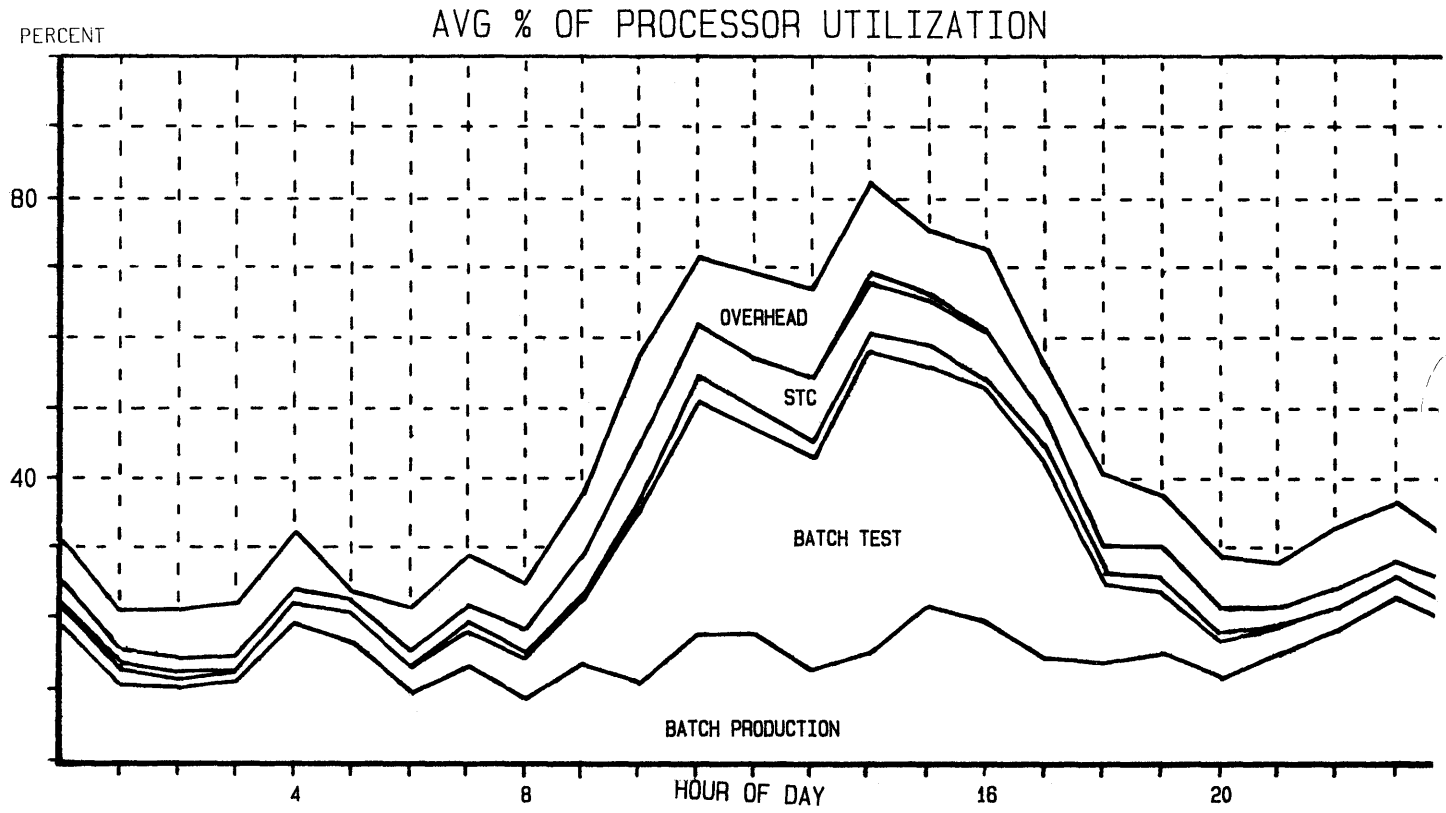


Figure 9.

PERFORMANCE MODELING

Parallel to the process of alternative generation, a model of the present computer systems environment should be built and calibrated. Each processor is modeled separately with the model reflecting current workload and utilization for these workloads.

After calibration, the workloads may be increased in 5% increments and the impact on service examined (See Fig. 8). The key service indicator for on-line services (TSO, CICS) is response time and for batch workloads it is relative elapsed time of the workload. In all cases, results are shown relative to today's service level. The modeling iterations can be done for the first 12-18 months of the planning period, depending on the growth curve. Steeper curves will introduce modeling inaccuracies earlier in the process. The models allow prediction of service levels for the early part of the plan period for each alternative. These become a key element in the formulation of a process recommendation.

FINANCIAL MODELING

In addition to service level projections, cost is the other element which has a major influence on the recommendation of an equipment strategy. All major equipment related cost elements (rent/lease, depreciation, maintenance, environmental cost, etc.) shall be factored into the cost model

Total costs for each alternative over the five year period are developed. Business realities dictate that both the total five year cost and the subsequent year budget cost are equally important in determining cost effectiveness. Both of these figures must be analyzed in determining the alternative to recommend to management.

RECOMMENDATION AND APPROVAL

Although the process of selecting the alternative is somewhat subjective, the availability of concrete data regarding service impact, cost and technological fitness, clarifies the decision making process. The next step is the presentation to senior management. Success can be achieved by structuring the presentation consistent with the steps in this paper. By beginning with business requirements, deriving the demand forecast, noting current status, service impact and cost for each alternative, and concluding with a recommendation, a direct link between business requirements and the equipment plan can be shown. Through this link, final approval by senior management can be achieved.

MONITORING

The development of a new five year plan is a yearly occurrence but the monitoring of the process is ongoing. The objective of the process is to ensure that the enterprise achieves expected levels of service. Performance is measured against service agreement goals each day; response time by type of transaction, on-time output delivery by individual report, availability of computing hardware, network and software components. The service is reviewed weekly with Senior EDP Management. Fig. 9 shows workload by category over a 24 hour monitoring period. Business drivers are analyzed monthly in terms of contracts, claims volumes, new products and regulations. The forecast of individual business indicators, transaction volumes and CPU utilization by application, are reviewed quarterly against actual data. Deviations from projections are explained.

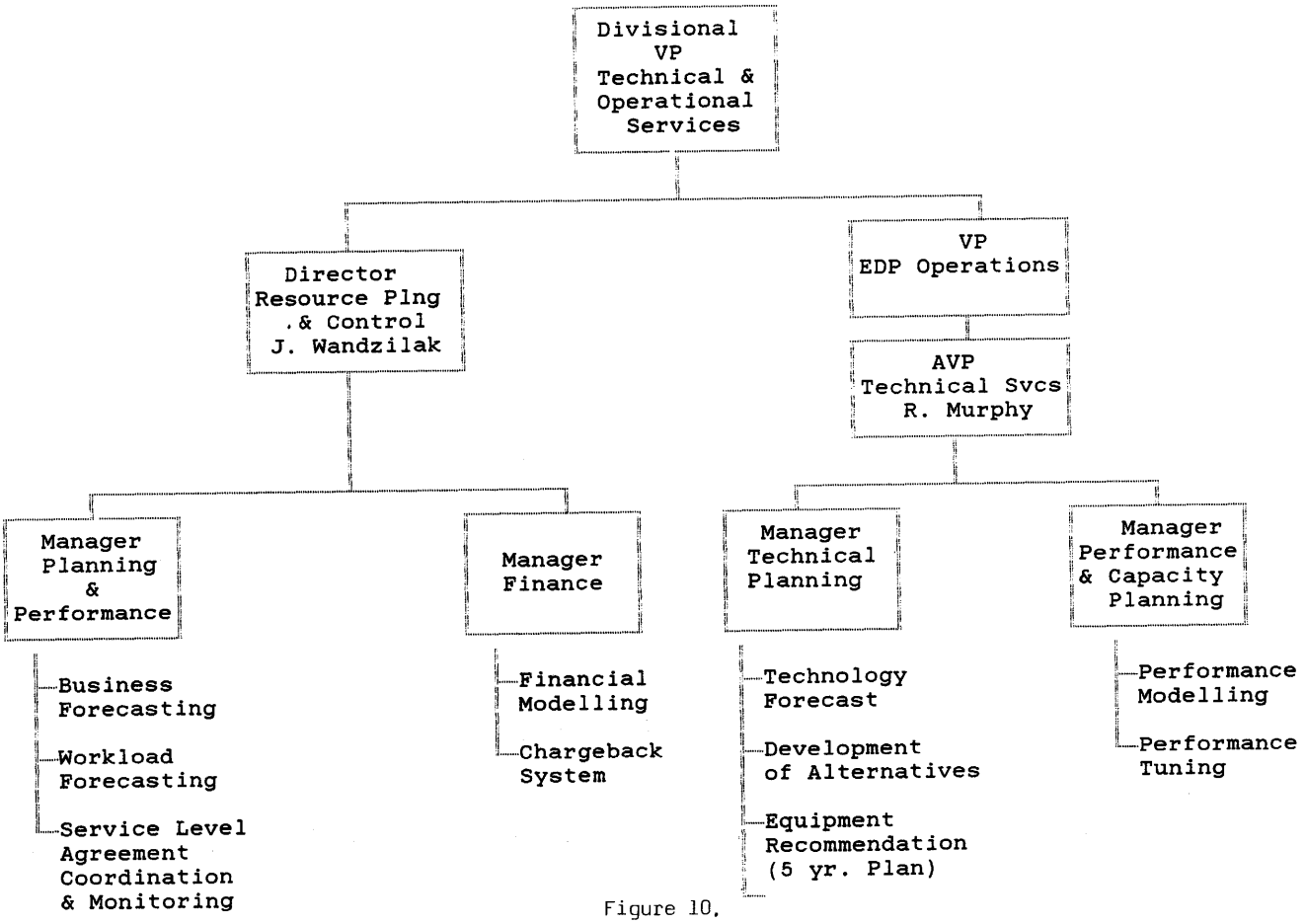


Figure 10.

ORGANIZATION & PERSONNEL

The organization of personnel to administer the aforementioned process is structured as shown in Fig. 10. The organization chart only shows the chain of command related to this paper, as are the functions identified next to the organizational unit. The responsibilities for the process are divided between the Resource Planning and Control group and the Technical Services area. In addition to the authors, approximately three analysts from each area are engaged in the computer resource projection process. The process starts in February and concludes before the July budgeting process for the following year.

RESULTS & SUMMARY

The results of our process for the past 4 years are shown in Figures 11 and 12. The business forecast, with claims volume as the major drivers, has been predicted within 3% of actual experience. The projections of On-line production transaction volumes have been within 5% of actual amounts. See Fig. 11. CPU hours are shown in bar graph form in Fig. 12 and show predications within a range of 1.5 to 9.5% of actual.

In summary, we have found a very detailed "grass roots" workload forecasting technique coupled with a rigorous performance and equipment capacity modeling process to yield dependable results for generating computer resource plans. We recommend this process to other planners not only for its reliability, but also as a technique to give management visibility to technical planning in business terms.

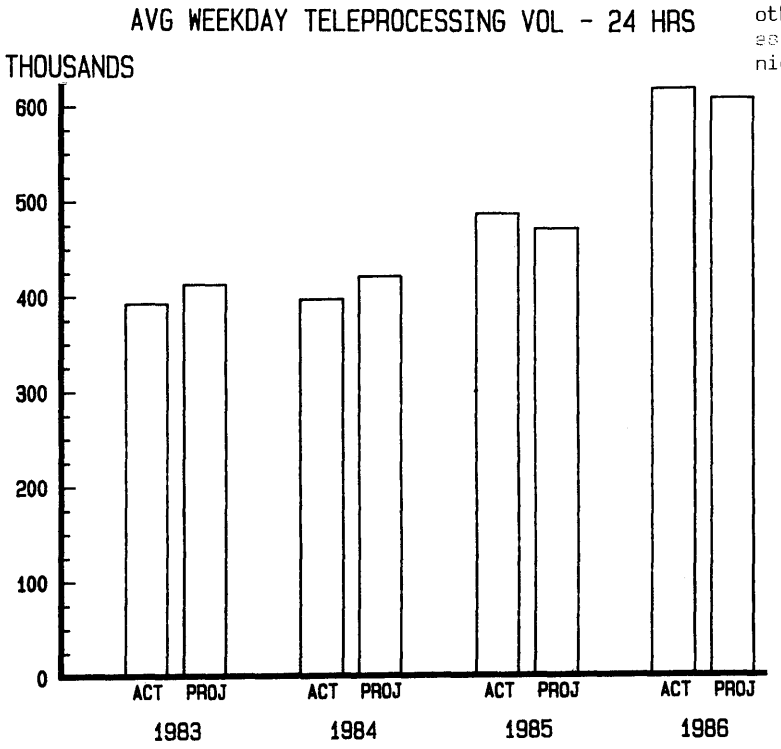


Figure 11.

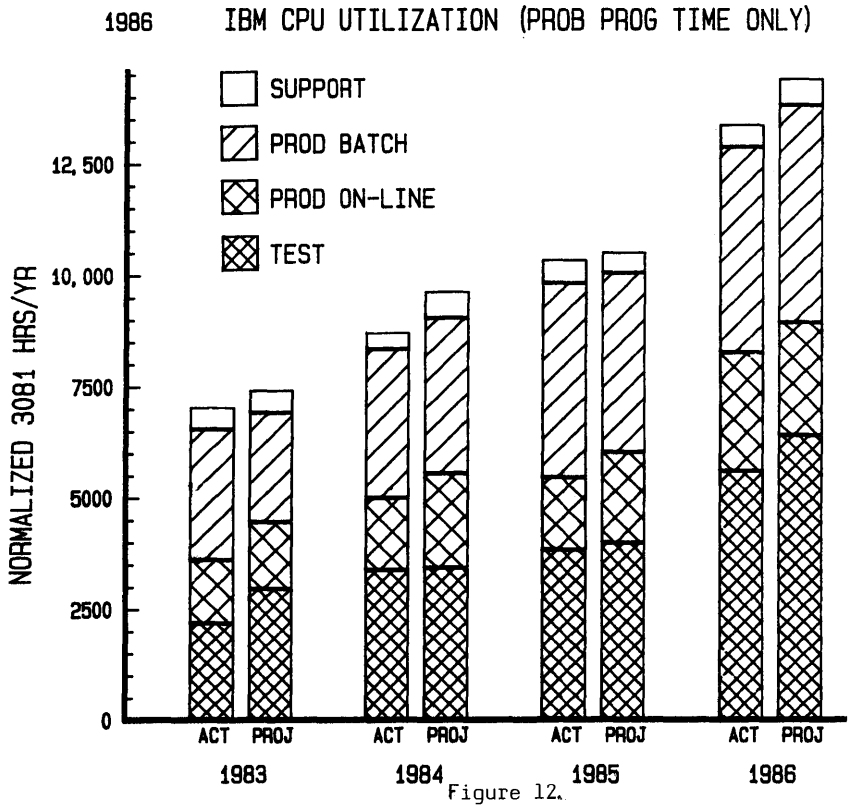


Figure 12.

CAPACITY PLANNING AS A BUSINESS PROCESS

Jerry L. Rosenberg
SRM Associates, Ltd.

ABSTRACT

It is becoming more apparent to capacity planners that they can no longer accomplish their jobs from the relative safety of the Data Processing community. Granted, it is less threatening to deal in jargon that we know, with people to whom we can easily relate, and in concepts with which we are readily conversant. That is exactly what we have historically done. But while we have been gathering data on I/O's per second and paging rates, data processing has grown into a major component of our enterprise. In fact, if management is totally honest, they will admit that the corporation can no longer function without data processing support. Because of this, corporate management is very concerned with understanding how they can best utilize "computing" to meet their goals and needs. Capacity planning should be the process that supplies these insights. This can only be accomplished if capacity planning is structured as a business process and not just a data processing function.

This paper will highlight the steps that should be taken to convert capacity planning into a viable corporate process. To accomplish this, we will be concerned with a method of defining the reasonable goals and expectations that the corporation should have for capacity planning. And just to be fair, we will review those goals and expectations that capacity planners should have of the corporation. These will include data requirements, time frames, support structures, and reporting needs. Of major impact will be the ever-present underlying requirement to understand and relate to the corporate "bottom line".

Introduction

Most Capacity Planners have accepted the fact that the process that they are engaged in must evolve into a business based function. This has been evidenced by discussions over the past several years concerning such items as Natural Forecasting Units (NFU) and Key Volume Indicators (KVI). It should be safe then to make the assumption that we are committed to posturing capacity management as a viable contributing component of the overall corporate entity. The author believes that any capacity planner that does not follow this path will find it increasingly more difficult to function effectively over the next few years. Given this first premise, it safe to also assume that we accept the requirement of understanding the rudiments of the industry in which our individual corporations are engaged. This knowledge should help us to identify those business elements of the industry that can effect the utilization of data processing resources.

These two issues: incorporating business concepts into capacity planning; and making capacity planning more responsive and useful to the corporation through the development of KVI's and NFU's, have been discussed over the past decade. The problem is that many of us are no closer to implementing them than we were ten years ago.

Since we have opened the door just a crack and let in some non-technical, non-data processing thoughts, perhaps it will not be considered too radical to suggest that we actually take a lesson from the business side of the company in structuring the foundation of a better capacity management process. In the strictest sense, data processing can be viewed as a business with a customer base and a deliverable product and/or service. This is equally true for the sub-component of data processing that we call capacity management. The establishment or advancement of this process should therefore be enhanced by the application of proven techniques from the general business community.

A Business Approach

In the sectors of the company that deal with the delivery of products to our ultimate customer base, there are definite guidelines for approaching the introduction of a new product or the major modification of an existing one. The first step is most often an analysis of the potential market to gauge the needs and interests of the customers. This concept of "market research", and its application to capacity planning, could help to supply us with some insights into the creation of a methodology that is more responsive to the needs of the corporation.

Define The Product

Any good market analysis must begin by establishing a concrete understanding of the product being offered. In our case, the service that is supplied by capacity planning is typically defined as:

The process required to ensure an environment with sufficient resources to deliver satisfactory service in a cost-effective manner.

There have been numerous volumes written on the expansion of this definition and the enumeration of all the component tasks that must be addressed, from workload characterization to management reporting. These issues are very important but are not the subject of this investigation. We are going to delve into this definition from the standpoint of what it can tell us about our potential market.

Define The Customer Base

In reviewing the definition, we can extract at least three specific areas of concern. "Sufficient resources" suggests CPU's, I/O devices, channels, network components, memory, etc. These are all within the province of the data processing departments of the company. "Satisfactory service" implies response times, transactions per hour, claims per day, accounts processed, customers serviced, etc. and relate to the end users of DP services. These are generally the operating departments that perform the actual functions of the company. "Cost-effective" is measured by return on investment, budget, fiscal strategy, etc. and are usually the concerns of the "executive branch" of the organization.

By reviewing the definition of the "product" that we are contemplating, we have identified three distinct customer types that we must satisfy. Each has a

different requirement from the process that we are developing. Graphically, we can consider this situation as follows:

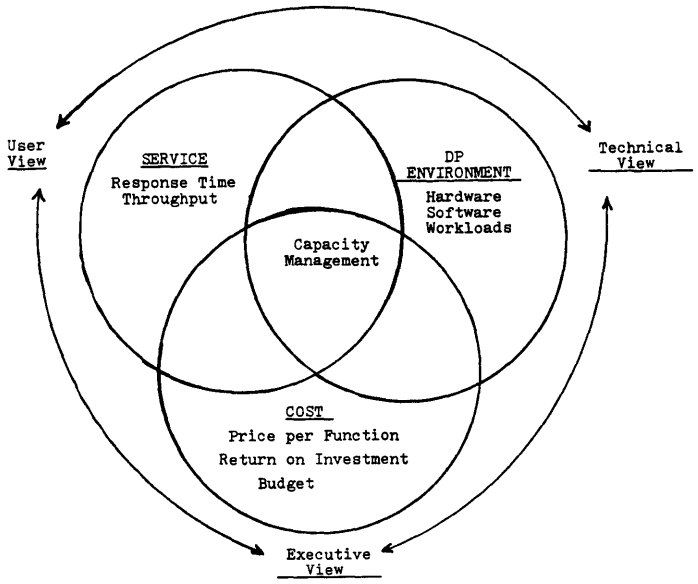


FIGURE 1

As seen, this illustration highlights three different views of capacity management; each with different needs, expectations, goals and resources. These are, in fact, the three major "customer types" for the capacity management process. In order to perform a market survey, we need to analyze each of these market segments, understand it in relation to the process and define the best approach to meet their singular and collective requirements.

Keep in mind that this is a very simplified view of reality. There are undoubtedly, in each corporation, many more subdivisions than these. For example, within the user community, there are executives as well who have different needs than line users. Also, within the data processing area we have executives and users in the form of TSO and CMS applications developers who need service to perform their jobs. Each list of "customers" will be slighty different. It should represent the clusters of homogeneous users who require capacity management support. For the sake of illustrating the process, we will work with these three broad catagories.

The Analysis

The "market research" activity should be viewed as an orderly progression that helps us to build a capacity management process that is useful to the organization by being beneficial to the segments that make up the corporation. In order to accomplish this task, we need only follow some simple steps:

1. Define the customer segments
2. Develop a "Concerns Inventory"
3. Expand these into a "Reduction Matrix"
4. Prioritize the resulting tasks
5. Proceed with the tasks

Since I am sure that this is not crystal clear at this point, let's explain each step.

1. Define the Customer Segments

As stated above, we must identify those groups in the corporation that require the services of capacity management. In the example, we will be considering the simplified case of a three segment environment - users, DP, and executives.

+ USERS

These are the people who depend on DP to provide them with services in support of their own corporate functions. They generally include a wide variety of business functions from payroll to engineering design. They all share some common requirements from data processing: accurate results that are available in a timely, reliable, cost-effective manner. They all desire as much service and functionality as possible.

+ DATA PROCESSING

This is the community that must establish and maintain the environment that supports the users in the manner that they require. In many instances, the "requirement" is actually different from the level "desired" by the users. As a result, one of the functions of the process will have to be the reconciliation of user desire to DP capability. In order to meet as many requests as possible, DP wants as much state-of-the-art hardware and software as it can get.

+ EXECUTIVES

This group is concerned with the overall goals of the corporation. They are faced with balancing the need for expanded capabilty and functionality against the budgetary bottom-line. They need to be convinced that expenditures will have a long-term return on investment.

These are the three segments that we will be addressing in our example. The danger in this approach of segregating customers is in falling prey to the belief that an adversary relationship needs to exist between these groups. This need not be the case. Unfortunately, it has become the status quo in many organizations. Our goal is to enhance the activities of the entire enterprise. Capacity management must be approached as a vehicle for helping each component work best in the common interest.

2. Develop a Concerns Inventory

This task requires the itemization of those detailed points that need to be researched within each customer segment to insure a sound process. While there may be some additions for specific environments, a reasonable starting point would include:

+ Expectations

What does each customer think he should be getting from the capacity management function? In a market survey for a new product, this activity would require the use of questionnaires and interviews. This same methodology should be utilized to obtain all of the information needed. It is impossible to develop a product to meet the needs of the community at large unless you find out what they really want.

For each segment, list those items that are critical to the performance of their respective functions. This task should yield some insight into the groups "measure of success". In short, how does this group know that it is doing well. This can be measured quantifiably in number of claims processed or items delivered or it can be more subjective, such as "met budgetary goals". In any case, it will help us to determine the "bottom line" for each user community and help us to develop a process that best serves these needs.

+ Inputs to the Process
Itemize the specific input sources needed in the capacity management process that are available from each segment. The information in this area should include the specific source, the type of data, its use in capacity planning, the storage medium (hardcopy, DASD, tape, word of mouth, etc.), and the time frames associated with data availability and updates.

+ Committments/Communications
The key personnel in each group should be identified and the level of committment required to insure cooperation from all parties must be clearly delineated. Without committment and cooperation, the capacity management effort is doomed to failure. Since the process is very dependent on a strong degree of inter-departmental cooperation, each segment must be "in step" during the establishment and maintenence of the process.

In order to insure the on-going viability of the methodology, the lines of communications should be clearly defined. An aid to understanding the "channels" is to obtain an up-to-date organizational chart.

+ Benefits/Opportunities
Remembering that we are conducting a business-like market study, let's continue the analogy to its likely conclusion. Eventually, we are going to have to sell our product. In order to ensure that we can accomplish this goal, it is highly advisable to list all of the potential benefits to be derived by our customers. The primary advantage is that we can eventually use this information to convince reluctant customers to subscribe to our process. It is often helpful to make people feel that they are active contributors to the development of the process. As a secondary dividend, in listing the benefits to users we might uncover some additional report or service that we can provide that enhances that value of the function. Even if we cannot easily include these capabilities now, they can be considered for later enhancements or additions.

+ Timeframes
For any issues that have a time dependency, notations must be made and related to the process. Late delivery is often worse than no delivery. It is also embarrassing to promise delivery of a report that requires data that is not available until after the report date. It is vital that the process be built around these time constraints.

+ Deliverables
As a result of producing entries in each of the above areas, we should be able to develop a list of expected outputs that meet the needs of the customer base. Effectively, we are distilling all the requests into a series of manageable end products.

+ Limitations
Also as result of the analysis of all the above, we should list all of the concerns that cannot be addressed at this time with reasons and ultimate solutions. This section will help to direct future activities and will also help you to answer the customer who asks "why not?".

* Summary of Matrix

It should be apparent that a series of interviews and questionnaires are required to accumulate the points that make up the matrix. This approach does not eliminate the need to talk to the customers. In fact it is dependent on it. What we hope this procedure will provide is a means of summarizing the acquired data into a manageable format.

It is often of interest for the analyst to anticipate the contents of the matrix before beginning the data acquisition phase. Comparing the initial guesses with the final results can be used as a measure of how well the capacity planner understands the business and the user community. Of course it is important not to allow our initial belief to cloud the objectivity required in the interview process.

3. Expand these into an "Analysis Matrix"

Given the results of a reasonable interview process, we can now use the matrix to reduce the data into summary format. At that point, we will be able to summarize each row and column to obtain a list of tasks.

In our particular example, the matrix will appear as follows:

	USERS	DP	EXECS	SUMMARY
Expectations				
Inputs				
Commitment				
Benefits				
Timeframes				
Deliverables				
Limitations				

As previously mentioned, this matrix is not all inclusive and should be used as a framework for customizing a set of concerns for each specific environment. Often the development of this matrix will help to uncover other unique concerns within your organization that should be specifically included and addressed. As an example, we will be filling in this matrix later in the paper to illustrate the process.

4. Prioritize the Resulting Tasks

Once the matrix has been developed and the data acquired has been summarized into it, a review of the right most column and the bottom two rows (Summary, Deliverables, and Limitations) should easily result in the production of a list of tasks. Essentially what we are doing is using the matrix format to simplify the analysis. By reviewing each row, we should be able to develop a summary for that key item. By doing the same for each column, a list of desires and requirements from each user group shoud evolve. The intersect of these processes should yield the two boxes at the lower right of the matrix that have the distilled information that we need.

The capacity planner will now be able to discern the relative priority of each task dependent, of course, on the appropriate scheme in vogue in your corporation. Some of the more common prioritization techniques are: Technical value, Monetary concerns, Delivery of service, The highest ranking person always gets his way, etc. In point of fact, the system in place is usually some combination of the above. Since I don't have enough time or paper to cover the topic of the political realities of the business world, all I can say is - be alert and best of luck with this one.

5. Proceed with the Tasks

Having a prioritized list of tasks should help in keeping the process on track. Before you proceed , it is a good idea to review this with some of the customers who supplied the information (particulary those identified in the Committment section). This allows us to verify that we correctly interpreted the data we received.

* Summary of the Process

The key rationale for utilizing this, or any other technique, is to establish and maintain a perspective of the task at hand. Following this approach will force the analyst to proceed in an orderly pattern through the steps of data collection, reduction, analysis and result formulation.

The technique employs a standard business approach of surveying users concerning critical issues to help produce the most effective product. In this case, the product is a capacity management process that will be most responsive to the business needs of the enterprise. In order to step through the matrix, the analyst will conduct meaningful discussions with other departments in the corporation that will make him aware of the business of the corporation. In addition, he will discover the goals and needs of computer users, critical time considerations, available data sources, financial concerns and political realities. If he absorbs only a portion of these items, he will be much more valuable in his function as a capcity planner.

Matrix Example

In order to best illustrate how this matrix format can help to summarize data from seemingly disparate customer types, we will complete the matrix already used as an example.

Let us assume that we have distributed questionnaires that asked for clarification of the issues that we identified on the matrix. As a result of the responses to these inquiries, we followed up with interviews of several key persons in each community.

Due to lack of a spread sheet format in the proceedings, we will view the matrix one column at a time.

	USERS
Expectations	+ Understanding of resource comsumption by activity. For example, what resources are needed to process a particular function? + Accurate measures of service. How do we report response and delivery? + Clear service level agreements
Inputs	+ Projected volumes + Business plans + Details of anticipated new applications
Commitment	This segment will contain a list of key personnel that would be interviewed in detail.
Benefits	+ Better service + Better information for service level negotiations + Insights into scheduling + Information for the budget process
Timeframes	+ Volume schedules + New application schedules + Output delivery times + Response time needs

Perhaps as a result of reviewing this column, we could summarize as follows:

Deliverables	+ Reports related to service level delivery and cost of service
Limitations	+ We need to acquire data related to: -Business plans -Application plans

Let's move on to the next column:

	DP
Expectations	+ Utilization breakdowns by worlkoad + Future hardward and software needs + Bottleneck determinations + Clear service level agreements
Inputs	+ Classical monitor data + Technical information + DP plans
Commitment	This segment will contain a list of key personnel that would be interviewed in detail.
Benefits	+ System tuning + Adequate resources to meet user needs + Tracking methodology
Timeframes	+ Peak periods + Due out constraints + Workload overlaps

Perhaps as a result of reviewing this column, we could summarize as follows:

Deliverables	+ Workload characterizations + Reports on: Service by Workload Bottlenecks Future h/w needs
Limitations	+ Data may not be available readily for specific subsystems, i.e., CICS

We're ready to view the last column:

	EXECS
Expectations	+ Breakdown by line of business + Cost of DP by type + Optimal solutions to DP service
Inputs	+ Corporate plans + New lines of business + Potential legislative impacts
Commitment	This segment will contain a list of key personnel that would be interviewed in detail.
Benefits	+ Cost effectiveness + Return on investment analysis + Budgetary insights
Timeframes	+ Schedules for corporate planning process + Budgetary timelines
Deliverables	+ Reports on: Cost of DP Future alternatives with costs
Limitations	+ We need access to corporate plans, budgetary issues, state of the industry data, etc.

As a result of these three columns, we might summarize this way:

	SUMMARY
Expectations	+ Need to develop requirements by workload with associated costs + Must address the area of NFU's and KVI's
Inputs	+ Data is required from SMF/RMF, other subsystem monitors (CICS, IMS, etc.) + In addition, we need non-monitor data related to business volumes and plans
Commitment	This segment will contain a list of key personnel that would be interviewed in detail.
Benefits	+ Here we would detail how the process could be sold in each area
Timeframes	+ Here we would summarize the time requirements specific to each area

Based on this sample matrix (which has been kept very brief for the sake of discussion), the following tasks might be listed:

- Establish procedures for collecting and analyzing existing utilization data;
- Investigate and implement a monitoring tool for the requisite subsystems (CICS, IMS, Data Base);
- Regularly acquire and review both corporate and user department planning documents;
- Develop the relationship between business elements and DP usage. Establish a "cost per element"; and
- Design and publish reports that address the delivery of service both currently and in the future. Develop models that are capable of anticipating future system performance.

This list is by no means complete. Each item can be expanded into many sub-components. It should, however, be clear that involving the "customers" in the development effort will allow for a process that is more sensitive to their needs. The basic benefits of following this approach are:

- The involvement of all areas of the corporation in the development of the capacity management process will help insure support in the endeavor;
- It supplies a format for reducing and managing volumes of data into a few pages; and
- It supports a methodology for highlighting the critical tasks that must be performed to make the process successful.

Perhaps this is a good time to suggest the use of another business technique. Before you invest a lot of time and effort in the process, use the concept of "test markets" to see how well your product will be received. Start small and use the reactions that you get to help improve the output.

General Comments

Since we are interested in bringing capacity management into the mainstream of "the business enterprise", it is important that we always conduct qurselves in a business-like manner. To achieve this goal here are some general rules of thumb to keeep in mind:

- Know Your Customers

In a broader sense, know what your company does for a living. Unless you have this ubderstanding, it will be very difficult to relate to the needs of each segment of the company. This may require that the analyst occassionally review trade publications and establish relationships with peers in non-DP functions within the organization.

Optimally, this will be a two way process in which you will learn about the "business" and will help to educate others in the value of DP. It is also important to understand the organizational structure and the corporate decision making process. This will assist the capacity planner in determining the best point in the company to present information for optimal results.

- Satisfy Your Customer

This is an age-old axiom in business which still holds true. In the process of knowing your customers , it is key that you gain insight into his needs and requirements. Success is tied to satisfying those needs. As a part of satisfying the customer, concerns such as timeliness, accuracy, reliability and cost must be addressed.

- Gain the Customer's Confidence Through Positive Achievements

Nothing succeeds like success. Take a positive attitude and approach to performing your tasks. Exert some effort in trying to help the organization derive the maximum utilization from available resources. Develop an effective reporting system. Suggest alternatives that eliminate bottlenecks. Prove that the capacity management function is worth the investment. And do it in a positive cooperative manner.

- Present Multiple Solutions

Whenever possible, present more than one alternative for consideration. This is especially important when dealing with higher levels of management. Each option should be clearly delineated with benefits and impacts, costs and timeframes. This will help to get others personally involved and committed to the resulting decision. (please read on)

- Take a Chance (within reason)

While we advise you to present multiple options, do not be afraid to support the one that you think is best. Also, if you feel strongly, present original ideas and solutions. Gambles which are based on sound foundations come to be known as investments.

- Establish Defined Plans

Never appear haphazard and disorganized. Keep your tasks in perspective and document each milestone. The matrix approach presented in this paper has this concept as one of its cornerstones.

- Make Decisions

Management is not impressed by indecisiveness. If you hope to advance, you must exhibit leadership qualities.

- Constantly Review

Always remember that any process left to its own will eventually atrophy. Set up a viable feedback system that points out the strengths and weaknesses of the process. Correct the weaknesses and publish the strengths. Continually improve the process.

Perhaps it is time that capacity planners "do as the Romans do" and use some of the tools of the trade and the guidelines that have been employed by other segments of the business community for years. At the very least, if we start investigating how businessmen think, we will have a better chance of understanding their needs and helping them to succeed. If we accomplish that, then we succeed.

General References

[1] Abowd, A., "Understanding Financial Issues in Computer Planning", CMG Proceedings, 1984.

[2] Allen, Dr. L., "The Compleat Capacity Planner", CMG Proceedings, 1983.

[3] Brown, D.E., "The Case for Capacity Management: A Presentation to Management", CMG Proceedings, 1984.

[4] Bylinsky,G. "EDP Managers Put on Business Suits", Fortune, November 1976, pp. 68-74.

[5] Cortada, J.W.,"Managing DP Hardware: Capacity Planning, Cost Justification, Availability, and Energy Management", Prentice Hall, Inc., Englewood Cliffs, N.J., 1983.

[6] Febish, G.J., "Data Processing: A Corporate Asset or Liability", CMG Proceedings, 1983.

[7] Ferreira, J., and J.F. Collins, Jr., "The Changing Role of the MIS Executive", Datamation, Vol. 25, No. 13, November 25, 1979.

[8] Ingrassia, F.J., "Climbing the Corporate D.P. Ladder", CMG Proceedings, 1985.

[9] Ives, B., and M.H. Olson, "Manager or Technician? The Nature of the Information System Manager's Job", MIS Quarterly, Vol. 5, No. 4, December 1981, pp. 49-62.

[10] Lo, T.L. and J. Peng, "Capacity Management: A Multi-relational Process", CMG Proceedings, 1985.

[11] Michel, M.J., "Organizational Structure for Effective Information Technology Planning", CMG Proceedings, 1982.

[12] Morino, M.M., "Establishing an IS Management System", CMG Proceedings, 1986.

[13] Nanjo, H., R. Choy, and C. Liu, "An Information System Cost Analysis Approach", CMG Proceedings, 1984.

[14] Nolan, R., "Plight of the EDP Manager", Harvard Business Review, 1973.

[15] Persson, J., "VM Capacity Planning: A Management Perspective", CMG Proceedings, 1983.

[16] Rosenberg, J.L. and E.M. Friedman, "Capacity Planning in a Decentralized Environment", CMG Proceedings, 1984.

[17] Rosenberg, J.L., "Effective Management Reporting", CMG Proceedings, 1985.

[18] Rosenberg, J.L. and E.M. Friedman, "Do I Really Need a 3090 System?", CMG Proceedings, 1986.

[19] Stanton, J.A., "Managing MVS Installations as a Business", CMG Proceedings, 1984.

[20] _____, "Becoming a Manager: The Transition from Technician to Manager", CMG Proceedings, 1985.

[21] Tartaglia, D., "Capacity Planning: Justifying the Need", CMG Proceedings, 1986.

[22] Vince, N. C., "Establishing a Capacity Planning Facility", Computer Performance, Vol 1, No. 1, June 1980, pp.41-48.

[23] Von Kinsbergen, J.E., "An Integrated System for Capacity Mangement", Journal of Capacity Management, Institute for Software Engineering, Vol. 1, No. 4, 1983.

A CAPACITY PLANNING METHODOLOGY THAT IS DESTINED TO SUCCEED

Ruu-Tian Lawrence
First Bank System Information Services, Inc.

ABSTRACT

This paper was written to share with you how you can successfully establish capacity management function in your organization. The key issues this paper address are:

- A capacity management methodology that is destined to succeed.
- The organizational savvy that planners need to be effective.

To end the paper I will show you a recent quarterly capacity plan I developed. Please feel free to adopt ideas that are of value to you. I hope you will enjoy the success that I have enjoyed.

1. INTRODUCTION

This paper was written with the intent of sharing my experience as a Capacity Planner in two St. Paul organizations over the past four and one-half years. I can proudly say that in both situations the capacity management methodology was implemented with great success.

However, success was not obtained without pain. The pain I experienced came while learning two valuable lessons. One lesson concerned capacity management methodology. The other concerned technical knowledge and organization savvy that planners need to be effective. Both lessons can help you to successfully establish the capacity management function in your organization as well.

It has been a decade since the capacity management concept was conceived and its importance generally recognized. We capacity planners have come a long way. Nowadays capacity management is a very common and prevalent discipline which can be found in most medium and large data centers. This fact is illustrated by the rapid growth of attendance in annual Computer Measurement Group national conferences. The capacity management function has transformed from an art form to more of a scientific discipline.

There are numerous software packages on the market today which claim to help capacity planners do a convincing job in planning future computing requirements for their organizations. All these scientific, user-friendly, and expensive tools can help planners to increase the precision of their forecasts if they are used appropriately. Higher precision may ultimately bolster planners' credibility. Yet we planners often place too much emphasis on the tools we need before the capacity management function has been soundly established. We fail to recognize that a sound capacity management methodology is what provides the foundation that a planner needs to succeed during the start-up phase. Attempting to use an expensive analytical model without this basic foundation will result in high capital investment with low return. In fact, a good planner can initially establish the capacity management function with great success without any analytical models.

2. LESSON ONE: THE METHODOLOGY

A sound capacity planning methodology is no more than talking to the right people, placing accountabilities where they belong, forecasting frequently, and taking full responsibility for the recommendations. It is absolutely essential for planners to accomplish all four tasks to make his methodology effective.

2.1 TALKING TO THE RIGHT PEOPLE

Good forecasters use good indicators. They search for strong indicators. They normally have a good understanding of where they can find good information to base their forecasts on. Good planners need to do the same thing. We planners are forecasters. We need to know where to get accurate and reliable information so efforts spent gathering and validating data can be minimized.

I have relied on three primary sources. The first source is the systems development staff. These managers, analysts, and programmers are normally the first line interface with users. Because of their combined knowledge of both business and technology, they can greatly help in assessing the impact of new or enhanced computer systems.

The second source I rely on heavily is corporate planners and marketing staff. They normally possess a wealth of knowledge of business seasonality, trends, customer behavior, product promotions, etc. With their assistance we often can identify and explain patterned usage variations. At times we can even explain odd, unique, and non-recurring behavior.

To give you an example: during 1985 for a period of one month, Automatic Teller Machine (ATM) transaction volume increased by 40%. I was puzzled by that fact. After talking with one of the marketers I realized a campaign had been started to encourage customers to use Instant Cash Cards by randomly offering cash prizes. This promotion resulted in a 40% increase in ATM transactions on the mainframe. Luckily the mainframe was not saturated and there was no negative impact on the service level. However, Had I known in advance of this upcoming promotion, I would have been better prepared to deal with its impact--especially if the mainframe had been close to saturation.

The last source of information usually comes from diligently reading internal communications and following market changes and business related issues, such as bills to be decided during legislative sessions, etc. We planners need to follow closely all the issues which may affect the way our organization does business.

In the banking industry for example, state reciprocal banking law allows banks to acquire banks in other states that have corresponding reciprocal agreements. In 1986 when the law passed in Minnesota, my organization was suddenly in the position of being able to

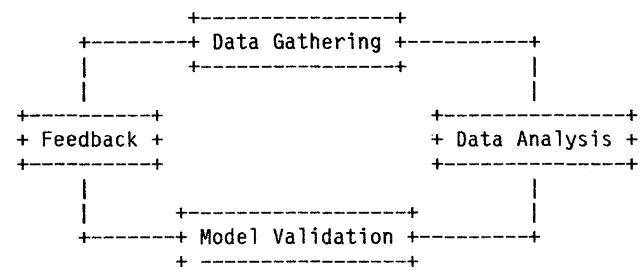
acquire a number of other banks. This could have instantly required increased computing resource requirements of many times our current capacity.

2.2 ACCOUNTABILITY

Capacity planners should not be the only ones responsible for the capacity plan. In order to explain why, I will start by looking at the objectives of capacity management and the planning process itself.

The objective of capacity planning is to assure adequate computing resources to meet future demands in a cost-effective way. The ultimate goal is to maintain the level of service which is agreed on or acceptable to the users. Service level objectives consist of availability, responsiveness, and reliability. Providing high quality and stable service in essence may increase employee productivity and maintain a good customer image. Since the impact of capacity management is corporate wide, to achieve its goal in essence is to achieve a corporate goal. Consequently, all employees need to share this responsibility.

The planning process itself consists of four major components. They are data gathering, data analysis, model validation, and feedback. It is a closed-loop, continuous process.



We all know the acronym 'GIGO' which means Garbage In Garbage Out. Accurate input information is critical to the reliability of the outcome.

Capacity planners, generally speaking, are not chartered to work with users directly on developing new systems or enhancing existing systems. Many times they are not kept current on all upcoming projects that may affect resource requirements. However, they are chartered to plan and predict future resource requirements for the organization. Therefore planners must get periodic updates on upcoming workloads to make the planning process complete.

As I indicated previously there are three sources from which I obtain information on upcoming workloads. They are the system development staff, marketers and corporate planners, and my own tracking of important business issues. In my organization, the system development staff provides by far the most information on medium term (six to eighteen months) resource requirements.

The systems development staff interface with users directly to define data processing needs and suggest ways to satisfy the needs. They most likely maintain and support some existing data processing applications for the users as well. They are close to their users and they best understand users' needs.

Frequently capacity planners need to actually beg for information on potential upcoming projects in order to forecast future resource requirements. This practice requires unnecessary effort on the part of capacity planners. The responsibility for ensuring that sufficient capacity exists to support new systems lies fully with the system developers who have promised a particular level of support to users. It is therefore imperative that the system development staff be accountable for providing projected requirements in a timely manner.

To carry this logic one step further, the organization probably should also hold the marketers and corporate planners accountable for providing information to capacity management staff and for working with them to derive accurate predictions. However, in reality this theory may not work. These groups are usually managed by a different division head whose responsibility is primarily outward focused. Their data processing expertise is often low. They normally do not share the same concern for data processing service levels.

Therefore, holding them accountable for providing input to the planning process may be impractical. There are other ways of obtaining this information which will be discussed in Lesson Two: Technical Knowledge and Organizational Savvy.

Capacity planners are responsible for developing the planning methodology. We are likewise responsible for marketing the methodology to management. We are responsible for convincing management that accountability in the process needs to be adequately placed. Capacity planners also need to assist our planning partners to better understand their current workload. By furnishing them with information on historical trends and resource consumptions for the systems under their responsibility, we will enable them to polish their forecasting skills and make them better forecasters. Competent planning partners, it goes without saying, give us a better chance to derive accurate forecasts.

By holding the system development staff accountable for input information, we already have them agreeing with the planning assumptions. By helping our planning partners to forecast better and by truly developing a partnership in the planning process, we planners will have allied ourselves with them all. When the final plan is presented, the capacity planners can assume there will be no one to challenge the validity of assumptions.

The planning process is a joint venture. The more planners that can assist, the better the result will be. The key is to offer our assistance and to work with the system development staff and other planning partners to create a win-win situation.

2.3 FREQUENT FORECASTS

Good forecasters forecast often. Where I worked, a quarterly planning cycle was adequate when supplemented by monthly comparison of actual versus forecast. A short planning cycle precludes us from being too far wrong. Periodic validating of our assumptions and predictions is extremely important due to the fact that plans and priorities inevitably change with time. When changes take place it is natural that predictions as a result should change accordingly.

However, when changes occur, it is critical for planners to be able to explain the difference between actual and prediction. All managers know that it is important either to operate within budget or be able to explain the variances. The same rule applies to planners. We need to either be accurate in the forecasts or be able to explain why the actual is different

from forecast. This is the only way we planners can establish our credibility.

2.4 BE RESPONSIBLE

Often I hear planners say, "My management is not committed." I challenge this type of statement. If management is not committed, why would they have a capacity planner on their payroll? I believe we planners need to be responsible for the capacity planning methodology. We need to market our ideas and prove the value of our existence. We need to make ourselves understood by our management. We need to focus on issues that our management is concerned about. We need to present our cases in ways that provide management all the information it needs to make decisions. We need to make management recognize our contribution to the organization.

In addition to our upward responsibility, we are responsible for working with people who provide us information that feeds into the plan. We are responsible for assisting them in forecasting better. More importantly we are responsible for giving them feedback. We need to convince them that the effort in seeking better forecasts is for the benefit of the organization. We need to convince them that we are there to help them look smart in their users' eyes. We are there to build a winning team.

3. LESSON TWO: TECHNICAL KNOWLEDGE AND ORGANIZATIONAL SAVVY

A good planner requires good people skills. Understanding bits and bytes is not enough. Unlike performance analysts, capacity planners have more exposure to the world at large. We cannot work in an ivory tower alone and expect to be successful.

For example, the problem I stated earlier about not being able to rely on outside departments for information can be overcome by a planner's organizational savvy. He can use informal influence to gather this information from corporate planners and marketers. Such individuals often cannot express the details of their service requirements in terms that are directly usable in the planning process. By providing them with historical trends of their application work volumes, the planner can equip them with the knowledge they need to quantify the impact of potential projects.

Of course, a planner needs to be technically proficient. He needs to understand operating systems. He needs to be fully aware of the integration of all applications. The level of application integration often determines the portability of certain applications from one mainframe to another. In addition to knowledge of operating systems, mainframes, and applications, the planner also needs to be trained in statistics. This knowledge will allow the planner to better understand the data. It will also enable him to better present the information, especially when the confidence of projection is not high due to future uncertainties.

On top of all the above knowledge, the planner needs to draw all the information around him as input to the planning process. He needs to have a very good understanding of the business he is in. The understanding of business equips him to better communicate with his users. The planner also needs to be politically sensitive. He needs to understand the issues that relate to the financial well being of his organization. Recommendations he makes should be technically sound and make good business sense. Additionally he needs to be able to present his recommendation persuasively and effectively. For example, graphical presentation should be considered due to its proven effectiveness.

In short, a good planner needs to be a technician, a statistician, a communicator and a manager who is capable of managing processes as well as people.

4. EXAMPLE: A RECENT CAPACITY PLAN

I would like to end this paper by showing a recent quarterly capacity plan which I developed. The plan has four basic sections. They are assumptions, volume projections, forecasts, and recommendations. Please feel free to adopt any ideas you feel would be valuable. However, try to be creative. Do develop a format that best suits your management's preferences.

4.1 CAPACITY PLAN FIRST QUARTER 1987
FIRST BANK SYSTEM INFORMATION SERVICES, INC.
JANUARY 1987

- The plan is developed to accommodate daily peak demand.
- 85% of utilization is used as processor (CPU) threshold. Higher usage may cause performance degradation, therefore, decrease our ability to maintain the established service objectives.
- Real storage and virtual storage are also taken into consideration.

CAPACITY PLANNING ASSUMPTIONS

RETAIL

- Lobby Automation
  Implementation will continue. The Metro banks will totally be implemented by July 1987. Five region banks will be implemented by December 1987.

  Add Credit Scoring feature. Will test through April 1987. Will implement in May 1987. May take in credit bureaus' credit rating in December 1987.

  To allow Otts transaction be posted to CICS directly. Test through April 1987. Will implement in April 1987.

  Large Currency Reporting system will be implemented through February 1987. Will be in full production in March 1987.
- CIS Expansion
  Incorporated Credit Card information in June 1987. Trust information in July 1987. Insurance information in October 1987. Discount Brokerage in October 1987. Mortgage in October 1987.
- First Milwaukee
  Will convert all retail systems by March 1987.
- Due From Reconciliation
  Initial implementation includes Fed. Reserve accounts for FBMP and FBSP. Will implement online by March 1987. Accounts for other banks (ex: Chase Manhattan) in June 1987. Will include regions in July 1987.
- 1099 Window Tran
  1099 window transaction volume peak at year end.

- 1099 Year End Correction System
  Automate 1099 year-end-correction process by January 1987.

- Collection Management System
  Will implement Credit Card by January 1987. In production in February 1987.

- IL and DDH Deconversion
  IL and DDH will remain IDMS applications throughout December 1987.

- First Mortgage
  Testing the Risk Management module in the Encompass system by May 1987. Implement in June 1987. In production in July 1987.

- Others (Impact NOT assessed)

  Account Reconciliation System
  Currently we are evaluating an ARS system from DISC corporation. Expecting to make a decision on software package by April 1987. Testing by May 1987, and implementation in June 1987.

  APEX System
  Two alternatives are now being considered. They are IBM and Tandem softwares. Will complete software recommendation by January 1987. Will complete hardware recommendation by March 1987. If it is approved, then will test in third quarter 1987; and be in production in fourth quarter 1987.

  Check Processing System
  We are currently evaluating the whole check processing system with SI. The result of the project may or may not impact IBM mainframes. Regardless the result, we do not expect to see any implementation plan until the fourth quarter 1987 at the earliest.

  Saving/CD's/Retirement System
  A study will be conducted to determine what all needs to be done to improve the savings, CD's, and retirement systems.

  Checking Plus
  Enhance to handle larger retail loans in second quarter of 1987. This may enable the move of retail loans from CLII to Checking Plus systems.

  CIF Files Consolidating/Marketing Analysis System
  Develop a consolidated FBS customers' file. This will allow personal bankers easy accesses to a complete customer business profile. Along with this effort, we are currently evaluating softwares which will facilitate cleaning and standardizing data elements process of CIF files. Providing householding feature will be part of the project. We are developing the POD now. Will test consolidation in March 1987. Implement the consolidation system in May 1987. Automatic clean-up by July 1987. Manual clean-up will be part of the routine process; and householding feature is expected to be implemented in July 1987.

  Unistatements
  Possibly add more types of accounts and more relationships to the system. For example: adding credit card accounts to unistatements. This will be a batch system. Target to implement in the third quarter 1987.

  Float Pricing System
  Possibly modify the existing Burroughs system to provide SI more flexibility in controlling float pricing. Expecting a decision by May 1987. Implementation in September 1987.

  Routing System
  Possibly replace the regional transmission (X300) with RJE jobs. The new system will handle the input as well as print-back capabilities. POD is completed in January 1987. If approved, will test through June 1987; and will implement in July 1987.

  Sales Management System - Retail
  Will develop a system to monitor and track retail sales in 1987. This is to support incentive compensation pay for personal bankers. Will implement in February 1987; and in production in March 1987.

  Sales Management System - Small Business
  Will develop a system to monitor and track small business sales in second quarter 1987. Implementation in third quarter 1987; and in production in third quarter 1987. Mainly a batch system.

WHOLESALE

- AIM System
  50 administration assistants are using online AIM system. This usage will continue to grow. Will start CIS Monetary balances project in March 1987. Will test through October 1987. Implement in November 1987.

  Plan to modify the system to reduce online transactions. Test through June 1987. Implement in July 1987.

  Will develop a product information sub-system. Will complete specification by January 1987. Test through May 1987. Implement first phase in June 1987; and implement second phase in October 1987. This is mainly a batch system.

- Trust System
  Test the security movement package by December 1987. Will implement in 1988.

- Stock transfer
  The system will leave us by March 1987.

- Due From Reconciliation
  Develop a Due Rec system for Commercial loans. Complete POD by January 1987. Will test through March 1987; and will implement in April 1987. It will be a batch system.

- Others (Impact NOT Assessed)

  AIM System
  Enhance AIM system to be the wholesale CIS system.

  AIM Junior Project
  Will eliminate the information stored in the system for Small Business for Minneapolis and St. Paul. Will test in April 1987. Implement in June 1987.

  Commercial Loan II
  Automation of foreign exposure risk analysis may happen. If go, will incorporate IBG, MIDAS information into CLII database. In the second phase, will incorporate DDA, CD's information into

the CLII database. Expecting a go or no-go decision by March 1987. Test through June 1987; and implementation in July 1987.

London Project
This project is currently on hold.

SYSTEM

- Network Services
  HCF facility will be installed in first quarter 1987. This facility will allow FBS Mortgage system 36's to communicate with each other. HCF along with reverse passthru will also allow HP devices to communicate with the IBM mainframes.

- Advanced Function Printing
  AFP is available for use. Complete implementation plan by January 1987. Will implement in IS by March 1987. Implement production print throughout 1987.

- System Software
  Enhancements to system software are required to improve availability, support new hardware or provide software for new and larger applications. Resources required for this support are included in the forecast.

- Others (Impact Not Assessed)

  Implementation of Single System Image is not quantified.

DIVESTITURE

- Sturgis
  Deconversion is completed in December 1986.

- Bremer Banks
  Deconversion is now in progress. Target to complete by December 1987.

- Others
  Status change from affiliate to Corr between August 1986 to July 1987. At this time, they will leave the HR, Payroll and G/L systems. Six months after the status change they will leave FBS system completely.

  Mengedoth Banks - Status change in January 1988. Deconvert six months after the status change.

ENTERPRISE BUSINESS

- G/L Business Line of Reporting
  Full production in February 1987.

- Management Report/Funds Transfer Pricing
  Develop funds transfer pricing module in October 1986. Eliminate existing management reporting system in January 1987.

- Others (Impact NOT Assessed)

  Human Resources System
  The project is progressing slowly. No impact is assessed on the IBM host in 1987.

OFFICE OF TECHNOLOGY

- End User Computing
  Focus
  Have approximately 60 focus users in December 1986. Will have another 110 focus users by December 1987.

  Issco Graphics
  FBSSI and Trust will pilot until December 1986. They will be in production in January 1987. Have approximately six users in December 1986. Will have additional 24 Issco graphic users by December 1987.

- Others (Impact NOT Assessed)

  End User Computing
  Issco Plan - This is a project management facility. FBSSI will be the initial user of this product. However, how the use of this product proliferates is an unknown at this point of time.

  Office Automation
  The possibility of implementing any office automation facility is unknown currently.

ONLINE TRANSACTION VOLUME FORECASTS
(Peak Hour Only)

New Application Growth	12/86	Addition by 12/87
Lobby Automation	3,500	5,000
AIF Expansion	-	5,200
First Milwaukee	-	3,000
Collection	900	1,000
FBS Mortgage	750	100
Due From Reconciliation	-	100
G/L Regional	2,100	500
G/L Metro	1,200	500
1099 Correction	-	50

Existing Application Growth

An average of 3% annual growth is assessed in the forecasts.

END USER COMPUTING AND OFFICE AUTOMATION
(Number of Users)*

	12/86	Addition by 12/87
Focus	60	110
Issco Graphics	6	24

* This indicates the number of users who have access to the systems. However, the real driver of the usage is the number of logon (i.e., active) users. In the analysis, we assume that one out of every seven end users is active during the peak hours.

GRAPHS LAYOUTS

Major components of the Graphs are:

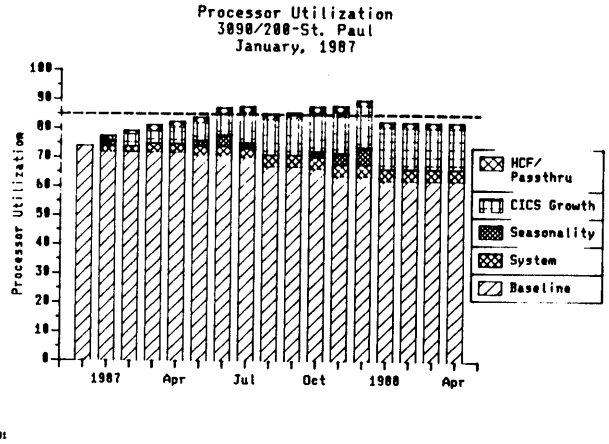
- Baseline - The processor utilization consumed by all existing workload which includes system and applications currently.

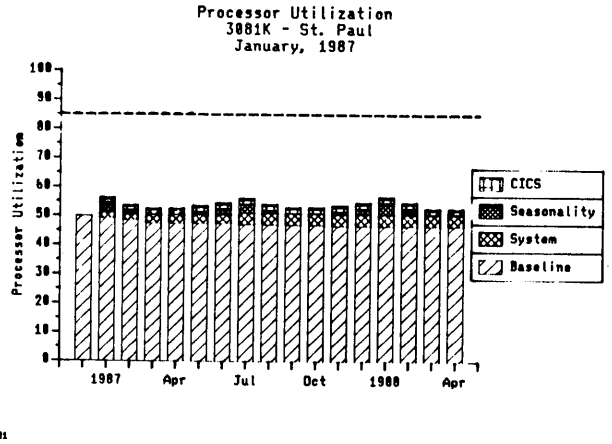
  Normally the declining baseline indicates the effect of divestitures of banks or applications.

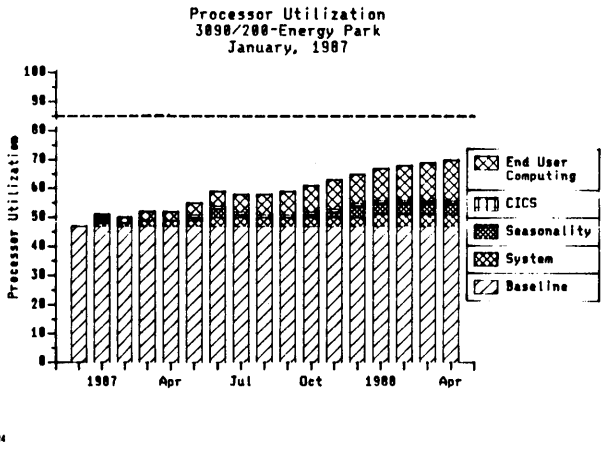
- System - Forecasted growth introduced by additional system utilities or enhanced versions of the existing system utilities.

- CICS Growth - Forecasted growth introduced by the new applications and the normal growth of the existing applications. However, new applications whose resource consumption are significant may be listed individually on the graphs.

- Seasonality - Forecasted additional usage due to the seasonal processing.
- Significant Events - New applications whose resource consumptions are of great significance are shown individually on the graphs.





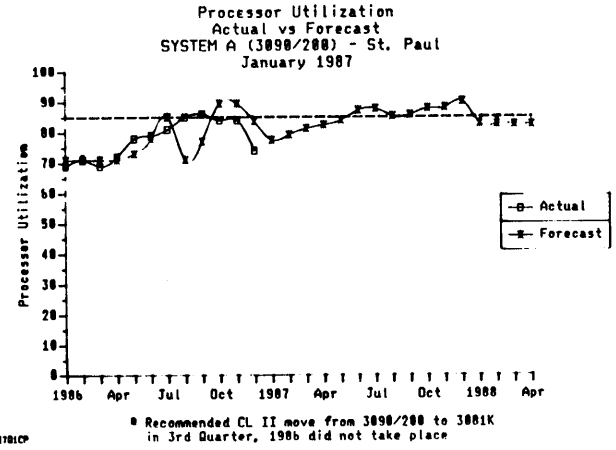


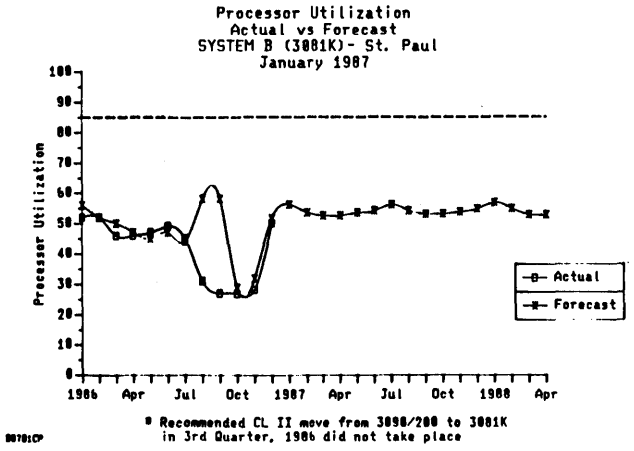
RECOMMENDATIONS

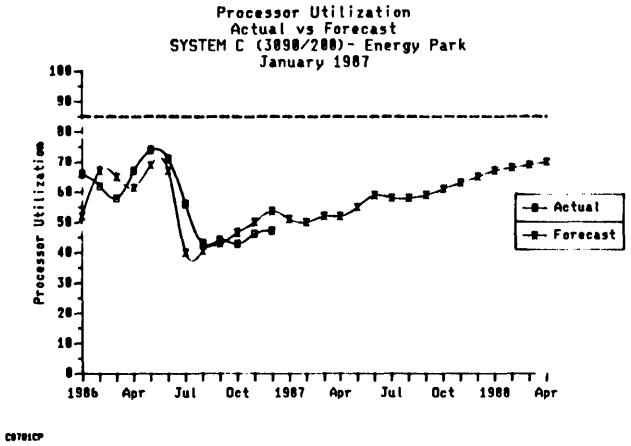
- If we implement new release of UCC General Ledger, two general ledger databases will merge in the first quarter of 1987. The merge will move the G/L database on the 3090/200 at St. Paul to the 3081K at St. Paul.
- Continue to explore the possibilities of implementing CICS Single System Image. This would allow users to access all three processors without being burdened to logon and logoff from one processor to the other.

  The Single System Image needs to be implemented by April 1987 either partially or in full. To alleviate capacity constraints in June 1987, we need to position to move Commercial Loan II and International Banking from 3090/200 to 3081K in St. Paul site and to move Personal Trust from 3081K to 3090/200 at St. Paul site.

  If Single System Image cannot be implemented by June 1987, then upgrade the 3090/200 to a 3090/400 at St. Paul site should be considered as an alternative.







4.2 DASD PLAN FIRST QUARTER 1987
FIRST BANK SYSTEM INFORMATION SERVICES, INC.
JANUARY 1987

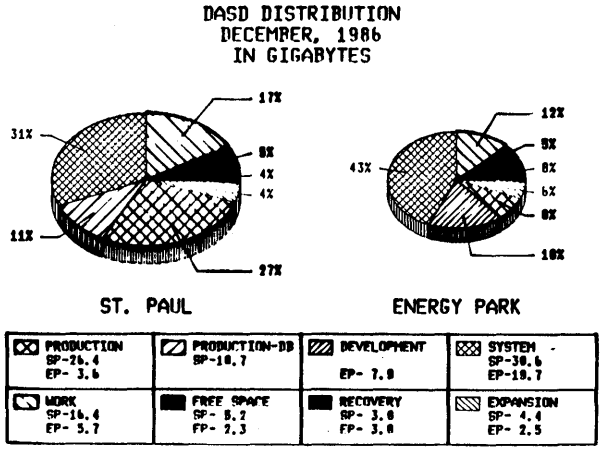
GRAPHS LAYOUT

- Production
  This is the disk space used by production systems which are non-IDMS.
- Production-Database
  This is the disk space used by IDMS production systems.
- Development
  This is the disk space used by application programmers for developing or maintaining systems.
- System
  This is the disk space used by Operating System, System Network facilities, program products installed in the systems, and some operational related files.
- Work
  This is the disk space required to support production and development work. Example: Sort space, temporary work space, etc.
- Free Space
  This is the space that is available to meet future or unexpected demands.
- Recovery
  This is the space required to do recovery in case of hardware malfunction.
- Expansion
  This is the space required on each logical units for potential file expansion and fragmentation.

  Generally speaking, the expansion spaces are small and fragmented. They can only be used to satisfy small demands. Disk management group always tries to minimize it. However, for planning purposes, we will treat Expansion as used space.

COMMENTS

- We are currently using 2.5 Giga Bytes as the threshold. This is equivalent to 4 3380D logical units.
- The Expansion Factor is 8% of all the production, development, and some system disk logical units.
- We are currently using 3.8 Giga Bytes for recovery. This is equivalent to 2 3380D and 2 3380E logical units.
- The Charts do include all the 3350 disk drives which are still in use. However, they will be phased out in February-March 1987 time frame. This is the result of the replacement of IBM 3350 disk drives with IBM 3380 disk drives project.



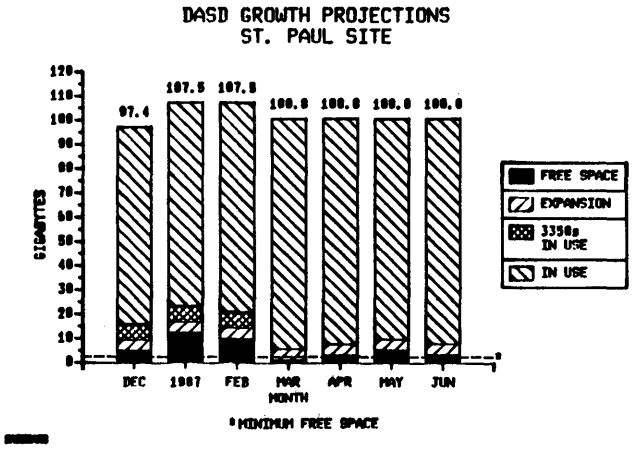
SUMMARY OF FUTURE DEMANDS

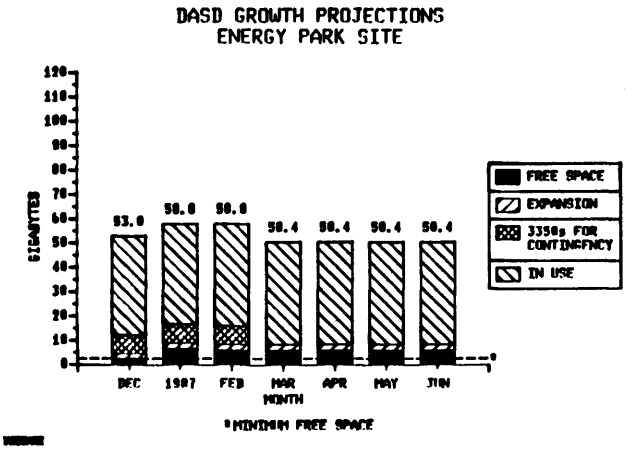
ST. PAUL SITE

Systems	Date Needed
• AIF File Expansion	1/87
• 1099 Correction System	1/87
• Mortgage File Expansion	1/87
• Collection Management	1/87
• Lobby Automation Support	1/87
• System/Database	2/87
• Billing File Expansion	2/87
• Human Resources System	2/87
• Personal Trust File Expansion	3/87
• Operation Print Project	3/87
• System/Database	3/87

ENERGY PARK SITE

Systems	Date Needed
• Human Resources - Focus	1/87
• System/Database	1/87
• Time Accounting System	1/87
• AIM File Expansion	2/87
• Focus/Gateway Expansion	2/87





WHAT WE CAN LEARN FROM OTHER CAPACITY PLANNERS: A SURVEY OF QUANTITATIVE METHODS

B. A. Ingraham
Pacific Bell

ABSTRACT

A variety of management science and operations research methods for planning under uncertainty or in complex situations are presented. The application of the methods to computer capacity planning is discussed.

1. INTRODUCTION

This paper is about a management philosophy in which quantitative methods are used to solve quantitative problems. This is NOT a tutorial which describes in detail how to implement quantitative techniques in specific situations.

The question "How is computer capacity planning carried out as a business function, rather than a technical function? ", has been raised repeatedly at the past few CMG conferences, in both presentations [3], and in the Executive Expectations of Capacity Planning panel discussions. At the root of this question lies one of the fundamental reasons why computer capacity planning has value to an enterprise. This reason is economics, the allocation of scarce resources to great or unlimited demands. Data processing competes for scarce resources, such as capital dollars, with other parts of the enterprise. If the other parts of the enterprise state their cases in business terms, and data processing states its case in technical data processing jargon, then data processing does not finish well in the competition.

Where can computer capacity planners learn to state a case in business terms? Capacity planning is not unique to computers and data processing. Capacity planning for the main productive systems is an indispensable business function in many large (and some not so large) enterprises, whether the output of the main productive systems is paint, insurance policies, or fast food. I refer to these non-DP capacity planners as "other capacity planners", for lack of a better choice of terms. (Some confusion is inevitable, as many computer capacity planners use the term "capacity planning" to mean "computer capacity planning".) Computer capacity planners can learn by studying other capacity planners to solve similar capacity planning problems with similar techniques instead of reinventing the wheel, or worse, expecting executives to learn data processing.

The paper is organized as follows: Section 2 looks at two different views of capacity planning. Section 3 examines where capacity planning falls on the planning continuum. In Section 4, a few important quantitative methods are discussed, namely forecasting, queueing theory, linear programming, and decision analysis. Implications for managing computer capacity planning are discussed in section 5.

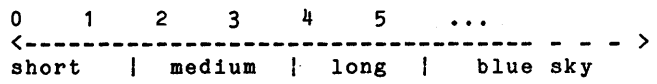
2. TWO PARADIGMS

There is a widely held belief that computer capacity planning is an outgrowth of systems programming and performance evaluation [12]. More specifically, it is held that experience as a systems programmer is a prerequisite to becoming a computer capacity planner, and that the tools used for performance evaluation are appropriate for solving computer capacity planning problems. Kune [13] calls such a set of shared beliefs about a discipline a paradigm. While the popularity of this paradigm attests to its success, the question posed in the introduction implies that there remain problems which are not solved to the satisfaction of executives making economic decisions.

There is another paradigm held by other capacity planners, in which capacity planning belongs to the business administration discipline of management of production and operations [9]. The tools of the trade in this paradigm are quantitative methods such as forecasting, decision theory, linear programming, and other management science techniques. I believe that we computer capacity planners can learn something from other capacity planners, in particular, how to apply quantitative methods within data processing to solve computer capacity planning problems. Seeing ourselves as business administrators rather than technicians will make the learning process easier.

3. THE PLANNING CONTINUUM

This section suggests a position for capacity planning on the planning continuum. The scope of planning can be anywhere from a few seconds to infinity. This is an awfully large range, so for practical purposes, it is helpful to break it up into smaller pieces. One way to split up the continuum is into tactical and strategic planning. Another way, which seems to fit the planning process better, is short, medium, and long.



The Planning Continuum

The scope of short-range planning is from the present to one or one and a half years out. The objective is to make detailed schedules for processing the work on the existing equipment. Medium-range planning covers the period from one or 1.5 years out to 3 or 5 years out. The objective here is to plan for enough equipment

to process the amount of work which will need to be done one to five years from the present. The focus is on aggregate planning, rather than details. The bulk of capacity planning, as a general business administration problem, ideally focuses on the middle range. Long range planning covers from 3 or 5 years and out. The objective is to make sure the physical facilities are in place to hold the equipment and people needed to process the amount of work which will need to be done 3 or 5 years from now and out. The sorts of things planned for are real estate and construction.

Note that as the planning horizon moves further out, complexity, uncertainty, the time available for analysis, the cost of mistakes, and the level of management approval all increase, while the degree of repetition decreases. For example, short-range planning may be done monthly, weekly, or even daily, medium-range planning is usually done at least yearly, but not more than quarterly, and long-range is ordinarily needed only on an ad hoc basis. All of these considerations effect the amount of time and money which will be spent on planning.

There must of course be communication among the short, medium and long range plans. I want to emphasize that the range of planning is a continuum, and there are big gray areas between short and medium, and between medium and long. When the time for capacity to be increased is near, then the aggregate and detailed plans must be reconciled. Capacity increases can sometimes be postponed by intensive efforts in short range planning. On the other end, the amount of computer hardware certainly effects the long-range plans for real estate and construction. However, a long-range capacity plan will not look much like a medium-range capacity plan. The problems, the solution techniques, and the people involved in long-range planning are all different from those in medium-range planning. The level of complexity, uncertainty, time available, and repetition in the medium range makes certain quantitative methods effective. The reason for this is, as we will see shortly, they were invented to help plan under uncertainty in complex situations.

4. QUANTITATIVE METHODS

This section will discuss some of the quantitative methods used for capacity planning. The list is by no means exhaustive, but is a survey intended to give a flavor of what is available and what it's good for.

4.1 FORECASTING

Forecasting is the foundation for all planning. Quantities which might be forecast in computer capacity planning include transactions per hour, CPU utilization, DASD space, tape drives, and print lines.

It appears to me that what many computer capacity planners expect from a forecast is one number for each forecast period, and absolute certainty that this one number will be correct. Many computer capacity planners also seem to fear that this is what management expects from forecasts. This is not a reasonable expectation. The future is uncertain. The goal of forecasting is to attempt to bound the uncertainty of the future as much as is reasonably possible. The bounds on the forecast are called X% confidence limits, and a forecast which includes X% confidence limits is called an interval forecast, as opposed to a point forecast. The value for X is typically 95. The interpretation of an interval forecast with 95% confidence limits is: based on past values, there is a 95% chance the actual future value will fall within the limits.

There are many forecasting techniques which will compute interval forecasts ([6], [10], [11]). They include stepwise autoregression, exponential smoothing, Holt-Winters, Box-Jenkins, statespace, and X11. The amount of time, skill, and historical data required varies with the method, as does the flexibility of the analysis and the quality of the forecast. The quality of the forecast also depends on the quality of the historical data: garbage in, garbage out.

A few comments about judging the quality of a forecast may be helpful. The accuracy of the forecast is not the only criterion. Confronted with two or more forecasts from which to choose, accuracy cannot be a criterion, since one would have to know the future already. Important criteria for evaluating models to use for a forecast are parsimony (the KISS principle), identifiability, goodness of fit, theoretical consistency, and lastly, accuracy [10].

In addition, a certain amount of sophistication is required when judging the accuracy of a forecast after the fact. Simply comparing the actuals with the forecast reveals little, since points are being compared with probability distributions. Techniques have been developed for testing whether the points fall where predicted [3].

4.2 QUEUEING THEORY

Queueing theory is an operations research method which relates waiting time in a queue to service time and server utilization ([2], [5]). It is used for determining numbers of bank tellers, studying runway congestion at airports, and for predicting the performance of computer systems. It has, based on the old computer capacity planning paradigm, been carried foreward as a computer capacity planning tool, especially for CPUs. The success of queueing models as performance predictors may encourage computer capacity planners to try out some of the other quantitative methods.

It is interesting to ponder whether queueing theory will continue to be as useful for CPU capacity planning as it has been in the past. A simple result from queueing theory, which is not discussed very much, is, as the service time decreases, the utilization at which the response time degrades approaches 100%. This means the shorter the service time, the higher you can drive the server utilization, whether the server is a CPU, I/O device, or anything else. There are three ways to reduce service time in a computer: optimize the application program, tune the system, and speed up the hardware. Great advances are being made in all three areas. At this point, it appears that current and future generations of large mainframes can be run at over 95% utilization without degrading response time [14]. No

additional information regarding the CPU can be gained from a queueing model, so the usefulness is diminished for planning for large mainframes. Queueing models are now more applicable as a computer capacity planning tool for slower CPUs, like superminis.

4.3 LINEAR PROGRAMMING

Linear programming is a constrained optimization technique with vast applications ([1], [5], [8], [9]). It is probably the best known and most widely used operations research method. The effectiveness of linear programming seems to come from its power to solve very large problems efficiently. In computer capacity planning, applications include DASD space planning, and the assignment of workloads to CPUs in multi-CPU data centers.

Linear programming was invented by George Dantzig in 1947, while working on a planning problem for the Air Force. I will let Dantzig speak for himself [1]:

> In a business setting one would hear, "Our goal is to make a profit." But you could never find any direct relationship between the stated goal and the actions to achieve the goal. If you looked closely at the next step, you would find some leader in his conceit had promulgated a bunch of ground rules to guide the way to the goal. This is a far cry from honestly looking at all alternative combinations of actions accross the board and picking the best combination. Those in charge often do a hand-wave and say, "I've considered all the alternatives," but this is so much garbage. They couldn't possibly look at all the possible combinations. Before 1947 the possibility that there could be a tool like linear programming that would enable one to examine millions of combinations was inconceivable. There was no algorithm or computational tool for doing so.

This sounds to me like the way a lot of computer capacity planning is still done today.

Linear programming turned out to be more effective than even Dantzig imagined. As an example, Joel Franklin [8] tells about his visit to a large oil company in 1958. The company had built a new multi-million dollar computer center. The computer was used mainly for linear programming, to make optimal production decisions which had previously been made by vice-presidents based on their experience. The computer center paid for itself in about two weeks. This gives some measure of the efficiency of linear programming compared with human judgment.

4.4 DECISION ANALYSIS

Decision analysis encompasses a number of methods for structuring the decision process ([5], [7], [9]). The techniques are helpful for planning under uncertainty and evaluating a large number of possible actions and outcomes. Some of the techniques are cost-benefit analysis, payoff tables, and decision trees. Among the quantities estimated and related in using these techniques are utility functions, payoffs, losses, risk behavior, expected value of perfect information, and the expected value of experimentation.

Here is a brief illustration. A decision tree may be used to structure the decision to buy a new CPU. Suppose there are three vendors to choose from. At the first decision level, there are four choices: buy from vendor 1, buy from vendor 2, buy from vendor 3, do not buy a CPU. For each vendor, there are several quantities to estimate: current price, price in the near future (Will there be a price cut? Is a VPA available?), new models available in the near future, reliability, life expectancy, salvage value, etc.. For the action of not buying now, some of the previous values are relevant, and in addition, the point of exhaustion, cost of degraded service, cost of additional tuning, etc., can be estimated. At the second decision level, the choices are, which of these estimates to make. The quantities involved at this level are the expected value of perfect information, the cost of each estimate, and so on. All of these possibilities and quantities can be structured on a decision tree, and the tree can then be "pruned" to find the decision with either the greatest expected payoff, or the least expected loss.

5. CONCLUSIONS

There are some obstacles to implementing quantitative methods in computer capacity planning. The first, not knowing they exist, I hope to have helped mitigate with this paper. The next is the required shift in paradigm, from seeing computer capacity planning as an extention of performance analysis, to seeing it as a subset of business administration. The time has come for us to remove our blinders so we can see what others are doing and learn from them.

There are also staffing and training considerations. If you are managing a capacity planning staff, you might begin steering your staff to take some training in quantitative methods. If you are hiring, you may want to consider someone with a background in management science or operations research, rather than systems programming or performance analysis. A good source of information may be the other capacity planners in the company you work for.

The potential payoffs are more efficient decisions built upon better forecasts and thorough analysis of alternatives. Computer capacity planning will be able to take advantage of the large body of knowledge concerning management science, operations research, and decision analysis, and thus avoid reinventing the solutions to problems which are already solved. Another benefit is computer capacity planners will communicate more clearly with business administrators, speaking in payoffs and costs rather than CPU seconds and megabytes. It seems clear from the last few CMG Conferences that this is the direction in which computer capacity planners want to go. A shift in paradigm and the use of quantitative methods are needed to establish computer capacity planning as an indispensable part of the business.

REFERENCES

[1] Albers, D. J., and Reid, C., "An Interview with George B. Dantzig: The Father of Linear Programming", The College Mathematics Journal, Vol. 17, No. 4, Sept. 1986

[2] Allen, A. O., Probability, Statistics, and Queueing Theory, New York, Academic Press, 1978

[3] Applegate, T. L., "Decision Rules for Forecast Tracking and Control: Living with Your Forecast as Time Goes By", CMG XIV Conference Proceedings, The Computer Measurement Group, 1983

[4] Artis, H. P., "The Five Stages of Capacity Planning", CMG '85 Conference Proceedings, The Computer Measurement Group, 1985

[5] Budnick, F. S., Mojena, R., and Vollmann, T. E., Principles of Operations Research for Management, Homewood, Ill., Irwin, 1977

[6] Chatfield, C., The Analysis of Time Series: An Introduction, 2nd ed., London, Chapman and Hall, 1980

[7] Einhorn, H. J., and Hogarth, R. M., "Decision Making: Going Foreward in Reverse", Harvard Business Review, Vol. 65, No. 1, Jan-Feb 1987

[8] Franklin, J., Methods of Mathematical Economics, New York, Springer-Verlag, 1980

[9] Gaither, N., Production and Operations Management, 2nd ed., New York, Dryden, 1984

[10] Harvey, A. C., The Economic Analysis of Time Series, Oxford, Philip Allan, 1981

[11] Ingraham, B. A., "Introduction to Box-Jenkins Forecasting", CMG '86 Conference Proceedings, The Computer Measurement Group, 1986

[12] Kolence, K. W., "A Personal View of the History and Future of Capacity Management", EDP Performance Review, Vol. 14, No. 12, Dec. 1986

[13] Kuhn, T. S., The Structure of Scientific Revolutions, 2nd ed., Chicago, University of Chicago, 1970

[14] Padamadam, C., "How to Get the Most Out of Your CPU", CMG '86 Conference Proceedings, The Computer Measurement Group, 1986

MANAGING DEPARTMENTAL MAINFRAME DEMAND THROUGH SHARED ACCOUNTABILITY

Kenneth C. Jones
GPU Service Corporation

ABSTRACT

As computer system usage has evolved from major applications developed and scheduled by an Information Systems function toward increasing client department development and scheduling of *ad hoc* programs and departmental systems, many organizations have felt they must better assist client departments in the effective management of their computer resource consumption.

Client departments and information centers have developed significant application and programming expertise as advanced languages make it easier to use computers without a technical understanding of the machines themselves. However, clients are less advanced in understanding the resources and costs required to support their activities. Thus, their use of computer resources sometimes seems to assume unlimited resources are available. At the same time, while other departments are retrenching they see IS continuing to expand – and do not realize that their own increased demands for automation are a key driving factor behind IS growth. It is therefore necessary for the technical experts within the IS function to translate computer resource consumption into terms which can be understood and effectively managed by the client community. IS has the data to support better management of this growth – now we must learn to market it to the rest of the corporation.

This paper reviews the techniques used by a mid-Atlantic utility firm to present information services usage and capacity data to non-IS management in a format which they can directly apply to managing their computer resource consumption. This reporting technique provides information by organizational and functional categories and uses non-IS terminology – dollars and names are provided, while technical measures are avoided for this audience. The reports have been very well received within the corporation, and continue to be expanded at the request of senior management.

WHY THIS PAPER?

Increasing development of computer skills throughout the corporation has led to greater demand for computer resources from outside the direct management scope of the Information Services (IS) function. As a result, there is a need for management outside IS to more actively manage their organizations' computer usage. This paper shares an approach to such management which has proved to be effective and well-received at General Public Utilities (GPU) Service Corporation.

Since the original development and distribution of the reports described here about 1-1/2 years ago, there has been a substantially increased awareness among non-IS management at our firm of the costs and sources of computer resource consumption. This has resulted in measurable changes in demand patterns, as well as in improved communications between IS and the rest of the company.

The reports are still growing and evolving at GPU, with the primary impetus for these changes coming from senior management – not from within the IS function. This paper was written to share our experiences with others in the field in the hope that they will be able to apply some of our lessons to their own organizations.

WHY ARE THESE TECHNIQUES NEEDED?

Increasing demand for computer resources which is driven directly by client departments – through *ad hoc* work in the batch, TSO[1], and/or CMS[1] environment – has resulted in a need to better manage this type of demand. This need is especially evident in the organization where this demand is a major – perhaps even **the** major – driving force requiring new capacity expenditures. However, IS cannot directly manage this growth since it is outside the scope of IS management control; thus, effective management requires communication and cooperation between IS and client departments.

In many organizations, the corporate image of IS professionals is that they are not a "real" part of the company. We are perceived as technical "wizards" who are remote from the real business of the organization. Our specialized skills are needed to provide computer services required to do the company's business – but for most of us, computing is **not** the company's primary business. Yet IS has had a historical tendency to grow at higher rates than the rest of the corporation.

Much of the image problem is due to poor communications. We must improve those communications and the level of cooperation between IS and the rest of the company. IS consumes a significant and growing portion of the corporate financial resources, and must be manageable. Ultimately, IS must do its share of the work necessary to understand the company's business and to help the rest of the company understand how their use of computer resources affects the major financial decisions made to provide those resources.

WHO ARE WE?

Before getting into the details of how we have addressed these requirements, the reader should understand the corporate structure in which these reports were developed. GPU Service Corporation exists as a separate corporation which provides centralized services for three operating electric utility companies under the overall GPU corporate structure. The Service Corporation functions include IS, materials & services, energy planning and finance, laboratory services, and similar functions which benefit from the economies of scale provided by centralization.

As a result of our location within the corporate structure, IS must be responsive to the needs both of the Service Corporation and of the operating utilities. A different corporate structure might require some adjustments to the details described below.

WHAT IS NEEDED TO IMPLEMENT THESE TECHNIQUES?

There are a number of types of data which must be available in order to implement this type of reporting within the corporation. First, it is essential to have information on computer resource usage at a detailed "who" and "when" level. In most organizations, this will come from a performance/capacity planning database (such as MICS[2]). For these reports, at least a month's worth of data must be maintained with enough detail to identify the individual responsible for a given session, job, etc.

In addition, these reports require corporate organizational data in the form of an "on-line organization chart" which will be used to group employees for reporting purposes. This information may be available through the security system, which frequently places userids into organizational groups to ease security definition.

There must be access to enough financial information to reasonably assign dollar costs to resource consumption. A chargeback system of some type is almost essential. Depending on corporate accounting practices, it may also be necessary to add some overheads to charges if they have not been included in the chargeback system. Without a chargeback system, it should still be possible to determine some estimated figures, "what chargeback would be", but this involves treading on rather shaky ground.

Finally, it will greatly improve the reports to be able to classify use into business applications such as "materials" or "customer accounting" rather than into IS applications such as "fast-class batch" or "TSO long transactions". When categorizing client-written code run interactively, the classifications should be understandable to the client as well, such as "graphics" or "SPF[1] edit/browse". If the ability to do business application classification is not available, consider developing it for better integration of IS into the corporation.

WHAT ARE THE FORMAT CONSIDERATIONS?

In order to have an impact with client department management, reports must be brief – one page per division, for example. Our reports list the top 40 clients on each page; the figure "40" was chosen because that is how many names will fit within the selected format.

It is important to develop a readable, attractive format. Because of the number of names per page, we chose not to use graphics in our reports. Graphics are appropriate to a large scale report, not a down-to-the-individual detail report. However, we did develop a boxed layout for ease of reading, and graphics are produced from some summary-level data. Remember, everything else being equal, a clean-looking report will have more impact than something that looks like "another computer listing". All the content in the world will not do any good if it is not accessible to the reader.

CHOICE OF REPORT TOPICS

The first point to remember when reports are generated is that the rest of the corporation expects and fears a "paper avalanche" from IS. Start small, and expand in response to feedback and requirements. Let the recipients adjust to one or two reports at a time.

Be careful to stick to topics for the report which can be understood and managed by client management. The client understands "12 job runs in the materials subsystem by J. A. Abernathy at an approximate cost of $294", not "12 BMPs accessing database family 381 submitted by userid K8934 with 46 minutes of CPU time and 125,000 database calls". Among the topics we have selected are job runs, TSO sessions and priority job executions.

Items selected for reporting must be manageable. The client cannot manage database calls, since it was IS that wrote the IMS[1] code. The client **can** manage transactions or jobs.

In addition to being manageable, the subjects must also be able to show an impact if they are better managed. In our 24-hour shop, the CPU is most highly utilized between 8 a.m. and 5 p.m. To maximize machine utilization, and return on investment, IS encourages shifting of use to evening and night hours whenever feasible. Our reports do not show usage during evening and night hours. At present, we do not really care who the non-prime users are – they do not drive our costs.

Above all, if the client cannot tell from your report how to manage the use – through names, applications, dollars, etc. – you may as well save the machine cycles and paper to produce it. The client cannot be accountable for managing things he cannot understand. Our job is to help him understand.

CHOICE OF REPORT TERMINOLOGY

This brings us to the overall key to any client reports which you expect to be used. If we continue to use "tech-speak", reports will have minimal impact.

The rest of the corporation manages in dollars. CPU time, main storage occupancy, I/O operations, database calls – none of these mean much outside IS. We need to translate into the language of business. Once we do that, client department management of dollars and sessions will result in savings in CPU, MSO, I/Os, and database calls.

In many organizations, dollars may simply be extracted from a chargeback system. At GPU, it was not quite that easy since computer chargeback figures enter the corporate accounting system and have a variety of overheads applied at summarized levels. It is not always easy to spread these back to the detailed reporting level.

If you have trouble assigning dollars, try anyway. When estimating, always do two things: (a) document the assumptions, and (b) make it clear to the recipient that this is an estimate. Bad dollar figures can destroy your credibility; no dollar figures **will** destroy your effectiveness. Assigning dollars may be difficult, but it must be done. The credibility problem may be ameliorated if you present early drafts as a first pass and invite feedback and corrections. Also, be sure that IS management has the first shot at reviewing the reports.

Determine what kind of dollars you are going to report, and make it clear what they are. For example, should the total match the IS hardware budget? the total IS budget? the total client bills? with or without corporate overheads? If you do not make clear what your figures can be compared to, they will be compared to everything – which means they will be wrong in every case but one.

It is also important to provide actual client names in as many places as possible. IS manages userids, the rest of the company manages people. In computer usage, there is a(n approximate) one-to-one correspondence. Do not make the client look that up.

Reports should be printed and summarized by corporate structure – division, department, etc. In this way, they can be targeted to the right management to see results. Incidentally, if your "on-line organization chart" has any errors, you will find out about it this way! Summarization by corporate unit

provides top-level management with visibility of the areas of highest, and perhaps inefficient, use. This is key in highlighting the areas where closer management attention can yield the biggest dividends.

GET TOP IS MANAGEMENT SUPPORT

Before going too far in developing this type of reporting project, obtain the involvement of top IS management. They will have input, much of which will help in those areas where technical people can tend to have a perspective problem. Their support is essential if you want to see results; as one of our managers would put it, "don't shoot uphill".

When the reports have been refined with input from top IS management, they will have their own "investment" in the project. At this point, they will want to sell it to the rest of the corporation. You need this involvement from IS management; they can sell it much better than you can.

APPLICATION OF THESE PRINCIPLES AT GPU

At GPU, we are still evolving in the process of applying these principles. We generate monthly TOP 40 reports, bundled by organizational unit and summarized at the overall company and corporate levels. We distribute the reports to the information centers, which work directly with the affected units in changing work habits where appropriate. We also send them to top management, for high visibility.

The actual reports we use are described below.

THE TOP 40 REPORTS

The first page is an overview (Figure 1) which shows how subsystems and corporate units share the total capacity of our installation. There was some technical unease in generating this, as we got into the details of sotware measurements vs. hardware measurements and so forth. However, what we publish is the best available **and it adds up.** As long as we are consistent in our technique, management has had no trouble buying into it. In fact, their perspective tends to be that they do not want to hear about the technical details, they just want a consistent report from month to month. We have documented our assumptions, and will notify all recipients of any changes.

We learned early that the percentages on this report must check fairly closely with all other reports for management acceptance. For example, if a corporate summary batch report is compared with a TSO report, the ratio for total CPU time must check with the overview page. This is true wherever it is reasonable to

XYZ SAMPLE CORPORATION
INFORMATION SERVICES
CLIENT COMMUNITY OVERVIEW

NOVEMBER 1987

========== MONTHLY TOTALS, 24 HOURS/7 DAYS ========= === PERCENT OF TOTAL CAPACITY, MONDAY-FRIDAY 8 A.M.-5 P.M. ===

---------- ON-LINE COMMUNITY ----------

SUBSYSTEM	REGISTERED CLIENTS	ACTIVE THIS MONTH	MAXIMUM CONCURRENT SESSIONS	OVERALL	ACCTG	SALES	MFG	ADMIN	DISTR	INFO SVCS
IMS-1	1,217	998	245	17.82%	3.71%	5.14%	1.47%	6.17%	.68%	.62%
IMS-2	978	694	314	10.11%	3.80%	3.35%	2.92%	-	.00%	.02%
IMS-TEST	619	582	42	4.19%	.05%	1.18%	.23%	.62%	.03%	2.06%
CLIENT TSO	2,098	1,053	151	10.77%	1.06%	2.81%	1.48%	3.50%	1.90%	-
IS/IC TSO	308	274	150	8.65%	.25%	.40%	.37%	.11%	.16%	7.32%
CICS	1,660	852	298	7.06%	1.09%	.17%	1.50%	1.69%	1.55%	1.03%
TOTAL ON-LINE	3,879	2,542	-	58.62%	9.99%	13.08%	8.00%	12.11%	4.34%	11.07%

---------- BATCH COMMUNITY ----------

SUBSYSTEM	REGISTERED JOBS/CLIENTS	ACTIVE JOBS/CLIENTS THIS MONTH	TOTAL JOB EXECUTIONS						
PROD BATCH	8,297	3,576	25,304	9.38%	2.80%	1.81%	1.45%	1.84%	.26%	1.19%
DEMAND BATCH	2,638	1,195	52,678	20.64%	1.38%	3.97%	.97%	5.80%	2.23%	6.27%
TOTAL BATCH	-	-	77,982	30.03%	4.18%	5.78%	2.43%	7.64%	2.50%	7.47%
* TOTAL *	-	-	-	88.65%	14.18%	18.86%	10.44%	19.76%	6.84%	18.55%

========== == MONTHLY PERCENT OF MEGABYTE-HOURS USED, 24 HOURS/7 DAYS ===

---------- DASD UTILIZATION ----------

DATASET TYPE	DATASETS	OVERALL	ACCTG	SALES	MFG	ADMIN	DISTR	INFO SVCS
PRODUCTION	27,186	17.19%	3.55%	3.87%	2.50%	3.13%	.08%	4.04%
CLIENT/DEPT ID	113,648	23.59%	1.76%	2.51%	1.65%	3.58%	2.38%	11.68%
PRODUCTION DATABASES	4,214	32.47%	8.83%	8.42%	6.22%	4.54%	.84%	3.59%
TESTING DATABASES	2,778	26.74%	6.64%	5.03%	4.84%	7.19%	.86%	2.16%
TOTAL	-	100.00%	20.79%	19.85%	15.22%	18.45%	4.17%	21.49%

NOTE: DASD SPACE USED FOR SYSTEM OVERHEAD HAS BEEN PRORATED INTO THE ABOVE CATEGORIES.

'-' INDICATES DATA IS NOT APPLICABLE.

Figure 1. Client Community Overview

NOVEMBER 1987
TOP 40 TSO CLIENTS FOR ADMINISTRATION
8 AM - 5 PM

RANK	CLIENT NAME	TOTAL SESSIONS	RAMIS	SAS	SPREAD-SHEET	GRAPHICS	SPEAKEZ	SPF EDIT & BROWSE	OTHER TRANS-ACTIONS	PERCENT OF ALL TSO	TOTAL CLIENT CPU HOURS	ESTIMATED MONTHLY COST	ESTIMATED YEAR-TO-DATE COST
1	RUSSELL, B.	37		3.2%		3.0%		3.2%	90.7%	0.7%	1.94	$1,544.30	$3,236.58
2	WEST, J.	62	100.0%							0.6%	1.71	$1,360.77	$4,008.42
3	ROBERTSON, O.	23	98.8%						1.2%	0.6%	1.66	$1,327.14	$1,638.35
4	BAYLOR, E.	41		1.5%				2.6%	96.0%	0.4%	1.23	$983.41	$1,813.92
5	CUNNINGHAM, B.	49	80.1%	6.3%				1.8%	11.7%	0.4%	1.20	$953.02	$2,953.37
6	HAIRSTON, H.	112		99.3%					0.7%	0.4%	1.07	$859.57	$3,087.62
7	REED, W.	69		7.1%				5.4%	87.4%	0.4%	1.04	$836.30	$2,208.18
8	BIRD, L.	57	0.1%	43.6%		19.7%		8.7%	27.8%	0.3%	0.82	$661.10	$2,356.45
9	THOMPSON, M.	37		42.3%		0.7%			57.0%	0.3%	0.78	$623.11	$738.69
10	RUSSELL, C.	28		42.4%		31.3%		7.6%	18.6%	0.3%	0.76	$609.25	$1,554.45
11	CHAMBERLAIN, W.	52	53.4%	2.4%				4.8%	39.4%	0.3%	0.72	$572.17	$679.98
12	LEMON, M.	51	22.9%			55.9%	0.1%	3.2%	17.9%	0.3%	0.71	$569.03	$1,854.49
13	MARAVICH, P.	53		60.3%		10.2%		5.8%	23.7%	0.3%	0.70	$558.92	$1,999.40
14	WILLIAMS, G.	53		0.1%		5.6%		2.8%	91.5%	0.3%	0.70	$557.29	$1,823.80
15	DREXLER, C.	36		3.0%		6.3%		3.0%	87.8%	0.2%	0.68	$542.06	$1,385.31
16	JORDAN, M.	49		29.7%		12.6%		12.0%	45.7%	0.2%	0.63	$506.86	$1,605.56
17	WILKINS, D.	34				0.3%		3.1%	96.6%	0.2%	0.62	$494.83	$833.67
18	CHAMBERS, T.	128		63.5%				7.4%	29.1%	0.2%	0.62	$494.32	$2,101.88
19	JOHNSON, E.	69		1.4%		1.1%		5.6%	91.8%	0.2%	0.61	$488.90	$1,973.53
20	OLAJUWON, A.	25		85.6%				3.3%	11.1%	0.2%	0.61	$487.82	$599.62
21	MALONE, M.	44		21.0%		9.6%		20.8%	48.6%	0.2%	0.58	$463.71	$1,726.44
22	PAXSON, J.	43		34.8%		8.6%		40.7%	15.9%	0.2%	0.55	$444.33	$1,605.12
23	ERVING, J.	43				29.3%		3.7%	67.0%	0.2%	0.54	$431.85	$1,834.59
24	GOODRICH, G.	73	77.6%	0.2%			0.1%	4.2%	17.9%	0.2%	0.52	$414.92	$639.19
25	SIKMA, J.	80		40.1%				13.5%	46.4%	0.2%	0.49	$391.63	$1,623.75
26	WALTON, B.	45					99.9%	0.1%		0.2%	0.48	$386.82	$675.87
27	LEVER, L.	43				0.8%		7.6%	91.6%	0.2%	0.47	$375.83	$1,263.40
28	RILEY, P.	44		10.5%		2.7%		10.1%	76.7%	0.2%	0.47	$374.53	$1,106.96
29	NELSON, D.	29	96.0%	0.6%				1.8%	1.7%	0.2%	0.47	$373.42	$805.87
30	JONES, K.	31		1.4%				19.8%	78.8%	0.2%	0.43	$347.04	$1,205.67
31	THOMAS, D.	46		0.3%		66.9%		6.2%	26.6%	0.2%	0.43	$342.63	$727.76
32	WILLIAMS, D.	51	95.9%						4.1%	0.1%	0.41	$330.39	$762.42
33	BROWN, F.	26		59.0%				13.0%	28.0%	0.1%	0.38	$305.74	$1,437.75
34	VANDEWEGHE, K.	24		3.7%				17.8%	78.5%	0.1%	0.38	$303.15	$1,237.38
35	SAMPSON, R.	21		3.5%				14.3%	82.3%	0.1%	0.37	$291.81	$2,104.89
36	BOWIE, S.	98	6.9%	18.7%				2.1%	72.4%	0.1%	0.37	$290.39	$2,766.38
37	WORTHY, J.	55		9.6%		4.8%		1.1%	84.5%	0.1%	0.36	$285.97	$1,574.33
38	MONROE, E.	66		29.8%				2.9%	67.3%	0.1%	0.35	$278.67	$485.23
39	WILKES, K.	40		0.4%		69.6%		1.1%	28.8%	0.1%	0.35	$275.55	$928.08
40	HAVLICEK, J.	24	92.9%					0.4%	6.7%	0.1%	0.34	$271.29	$849.35
ALL	160 ACTIVE CLIENTS	2,276	16.4%	17.5%	0.4%	7.8%	3.1%	6.6%	48.2%	18.8%	51.8	$41,410.75	$143,273.54

Figure 2. TOP 40 TSO Clients by Department

NOVEMBER 1987
40 MOST USED TSO FUNCTIONS FOR NEW WORLDS DIVISION
8 AM - 5 PM

RANK	FUNCTION NAME	NUMBER OF EXECUTNS	PERCENT OF ALL TSO	CPU HOURS	ESTIMATED MONTHLY COST	ESTIMATED YEAR-TO-DATE COST	#1 CLIENT WITHIN DIVISION		#2 CLIENT WITHIN DIVISION	
							CLIENT NAME	PERCENT OF DIV	CLIENT NAME	PERCENT OF DIV
1	QUEUE	53,934	6.5%	27.16	$15,614.60	$114,601.42	LEWIS, C. S.	5.0%	BURNETT, F. H.	1.6%
2	SPF EDIT/BROWSE	91,225	5.1%	21.21	$12,196.65	$77,503.34	WRIGHTSON, P.	2.0%	LUENN, N.	1.7%
3	TESTCOB	1,293	1.4%	5.92	$3,402.15	$18,087.38	TOLKIEN, J. R. R.	31.2%	BAUM, L. F.	12.4%
4	SAS	1,668	1.1%	4.54	$2,611.56	$11,113.88	SPYRI, J.	46.8%	STEVENSON, R. L.	17.1%
5	DB2/QMF	8,353	1.1%	4.48	$2,578.34	$13,238.79	TWAIN, M.	25.8%	L'ENGLE, M.	10.3%
6	GICS	16,906	1.0%	4.04	$2,320.20	$5,848.29	PEET, B.	60.1%	KIPLING, R.	14.8%
7	SCRIPT/DCF/DMF	12,756	0.9%	3.83	$2,204.56	$15,231.73	ALEXANDER, L.	5.3%	NESBIT, E.	5.2%
8	ASM2	4,615	0.7%	3.12	$1,795.46	$11,722.87	DOYLE, A. C.	10.8%	ANDERSON, H. C.	6.9%
9	FILE AID	3,364	0.7%	3.11	$1,789.25	$9,498.49	GEISEL, T.	43.4%	MACDONALD, G.	2.6%
10	SUBMIT	42,369	0.7%	3.02	$1,738.03	$12,365.02	GOUDGE, E.	2.6%	COOPER, S.	2.2%
11	RAMIS	2,666	0.7%	2.82	$1,622.80	$12,323.73	BURNETT, F. H.	16.5%	KIPLING, R.	12.7%
12	FMIS	790	0.7%	2.81	$1,617.39	$4,502.89	CARROLL, L.	52.9%	PEET, B.	21.6%
13	GRAPHICS	1,172	0.6%	2.36	$1,359.22	$10,119.14	TWAIN, M.	25.2%	BAUM, L. F.	21.0%
14	ENQ/DEQ	4,096	0.4%	1.72	$989.78	$6,907.98	COOPER, S.	31.1%	LUENN, N.	14.5%
15	ADF	5,559	0.4%	1.71	$985.70	$6,486.58	GEISEL, T.	9.8%	WRIGHTSON, P.	8.2%
16	IEBGENER	5,250	0.2%	0.78	$447.98	$3,168.55	THURBER, J.	16.2%	STEVENSON, R. L.	9.5%
17	TEST IMS	481	0.2%	0.67	$384.36	$1,980.54	COOPER, J. F.	38.3%	ALEXANDER, L.	13.9%
18	SECURITY	512	0.2%	0.66	$379.33	$1,772.36	DOYLE, A. C.	91.3%	DICKENS, C.	6.5%
19	DSPRINT	4,615	0.1%	0.55	$315.64	$2,484.41	TOLKIEN, J. R. R.	3.9%	GOUDGE, E.	3.4%
20	CLRSCRN	41,182	0.1%	0.49	$284.53	$2,078.67	BURNETT, F. H.	9.5%	KIPLING, R.	4.3%
21	EASYTREV	1,909	0.1%	0.47	$269.94	$3,394.49	EAGER, E.	13.0%	MACDONALD, G.	8.5%
22	INFOSYS	346	0.1%	0.37	$212.45	$966.03	L'ENGLE, M.	27.3%	SENDAK, M.	7.7%
23	SPEAKEZ	150	0.1%	0.24	$137.96	$1,922.79	DICKENS, C.	59.7%	ALCOTT, L. M.	16.1%
ALL FUNCTIONS			31.5%	131.7	$75,729.74	$524,519.64				

Figure 3. 40 Most Used TSO Functions by Division

compare figures. The fastest way to lose credibility is to have figures which do not check.

Our next reports focus on TSO consumption, which has been our fastest growing workload. One is by client (Figure 2); it uses such key concepts as client names (not userids), dollars as well as CPU time, and business category breakdowns. The other breaks down usage by function (Figure 3), so that a manager can see what his personnel are doing in a more detailed fashion.

We also have a demand batch report (Figure 4). In our environment, "demand batch" is defined to be all batch which is run at the request of a TSO session; this includes non-scheduled production work as well as code written by the client. This definition was selected because resource requests can be related to who "demands" it for management control. It is difficult to categorize this work into business functions, since much of the code is client-written. In this case, the client should be able to relate to a software product name such as RAMIS[3] or SAS[4], or to a category such as graphics or spreadsheet.

We produce a scheduled batch report as well, but it is the least-used of the set. In our environment, this cannot presently be broken down to the department level, which makes it harder to manage. We provide a job description, which helps in management. This was an area where some descriptions were inaccurate; our reporting of them led to corrections and better documentation standards compliance. On this report, the key control factor is number of runs, since run frequency is the manageable variable.

Another report we generate covers DASD occupancy (Figure 5). DASD is a very expensive resource, with total installation costs approaching the costs for mainframes. At the same time, it tends to be undermanaged throughout the industry, as the attendance at CMG DASD management sessions implies – everyone is looking for the key.

Our report is by client name, with the measurement and control variable a function of space occupied and time occupied. We intend to investigate the impact of time-of-day on this report. Would it make sense to convert this to a prime-shift-only report as we do with CPU-based reports?

We have also added a priority batch report to show the biggest users of priorities within the installation. This identifies clients and departments which most frequently bump themselves to the head of the queue. It has become somewhat controversial at GPU, since it allows each department to see how others are using priority. It brings the "priority wars" out in the open!

Our final report summarizes the dollar figures on the earlier reports (Figure 6). Management finds this to be very useful in identifying those clients with an overall high consumption pattern.

We conclude the report with a client index, showing the organizational information used for every client listed. This was added at the request of the receiving management.

NOVEMBER 1987
TOP 40 DEMAND BATCH CLIENTS FOR SUMMIT GROUP
8AM - 5PM

RANK	CLIENT NAME	TOTAL JOBS	RAMIS	IMS	EASYTREV	RETRAN & RELAP	PROJECT2	OTHER PROGRAMS	PCT OF DEMAND BATCH	TOTAL CLIENT CPU HOURS	ESTIMATED MONTHLY COST	ESTIMATED YEAR-TO-DATE COST
1	MALLORY, G.	27				0.1%		99.9%	1.6%	13.57	$7,801.74	$7,801.74
2	ROWELL, G.	179				94.6%		5.4%	1.2%	9.91	$5,700.31	$24,892.23
3	MUIR, J.	26				20.9%		79.1%	1.0%	8.88	$5,105.79	$21,101.59
4	KING, C.	99		83.4%	8.4%			8.2%	1.0%	8.61	$4,950.14	$36,195.52
5	HILLARY, E.	43				0.0%		100.0%	1.0%	8.57	$4,926.29	$7,559.60
6	LECONTE, J.	60				32.6%		67.4%	1.0%	8.29	$4,767.99	$56,776.76
7	MATTHES, F.	71						100.0%	1.0%	8.28	$4,759.84	$7,706.37
8	JONES, C.	70	100.0%						0.9%	7.30	$4,195.64	$5,743.33
9	ROBINSON, B.	17				100.0%			0.6%	5.42	$3,115.14	$10,072.49
10	FINK, R. S.	14	99.5%		0.0%			0.4%	0.6%	5.27	$3,027.63	$27,102.64
11	IRVINE, R.	93		55.6%				44.4%	0.6%	5.15	$2,960.29	$29,887.94
12	CLYDE, N.	319					97.9%	2.1%	0.5%	4.67	$2,685.36	$25,731.92
13	DYER, J.	29						100.0%	0.5%	4.63	$2,662.64	$6,778.28
14	BEDAYN, R.	21	4.8%	32.4%	15.7%			47.2%	0.5%	4.62	$2,654.88	$9,123.64
15	UNSOELD, W.	124				99.0%		1.0%	0.5%	4.31	$2,480.26	$4,077.56
16	WHYMPER, E.	45	39.6%	60.1%				0.2%	0.5%	4.26	$2,447.78	$4,430.24
17	DAWSON, G.	27	100.0%						0.5%	4.04	$2,322.89	$7,970.60
18	WIESSNER, F.	4						100.0%	0.5%	3.86	$2,216.76	$2,219.44
19	VOGE, H.	43		100.0%					0.4%	3.82	$2,194.43	$5,222.25
20	STARR, W.	41	79.9%	19.5%	0.2%			0.5%	0.4%	3.79	$2,177.25	$10,801.31
21	BECKEY, F.	30	82.5%	13.8%	3.6%			0.0%	0.4%	3.53	$2,032.04	$3,770.09
22	HINTERSTOISSER,	49	27.3%	67.5%	3.5%			1.7%	0.4%	3.44	$1,980.24	$13,656.70
23	EICHORN, J.	13		99.8%				0.2%	0.4%	3.41	$1,962.11	$2,427.27
24	GARDINER, J.	30	100.0%					0.0%	0.4%	3.05	$1,754.97	$9,666.01
25	MICHAEL, C.	22	97.0%					3.0%	0.3%	2.76	$1,585.56	$8,962.48
26	URQUHART, C.	48	53.0%	31.2%	2.6%			13.2%	0.3%	2.74	$1,575.79	$10,628.02
27	MENDENHALL, J.	52	98.4%					1.6%	0.3%	2.51	$1,440.59	$3,310.56
28	RICE, W.	34	28.8%	51.8%	19.0%			0.3%	0.3%	2.45	$1,409.74	$7,957.42
29	ADAMS, A.	18	59.9%	40.1%				0.0%	0.3%	2.33	$1,340.46	$5,566.15
30	SPAULDING, F.	58	9.7%					90.3%	0.3%	2.25	$1,294.37	$7,209.86
31	POWELL, J.	306					99.4%	0.6%	0.3%	2.25	$1,291.50	$3,824.21
32	MONTANO, F.	47	99.7%	0.2%				0.1%	0.2%	2.05	$1,178.42	$8,323.12
33	OUTRAM, J.	220					100.0%		0.2%	1.87	$1,075.97	$1,858.87
34	CHOUINARD, Y.	192					100.0%		0.2%	1.86	$1,069.38	$7,267.17
35	UNDERHILL, R.	25	91.2%	1.7%				7.0%	0.2%	1.74	$999.98	$3,854.22
36	WASHBURN, B.	217					87.8%	12.2%	0.2%	1.69	$972.22	$9,786.50
37	WHEELER, G.	81					100.0%		0.2%	1.68	$965.63	$4,322.35
38	OWEN, W.	15	100.0%						0.2%	1.67	$958.86	$3,799.81
39	REIGELHUTH, J.	66						100.0%	0.2%	1.61	$925.04	$8,373.49
40	JONES, K.	25	99.1%	0.5%				0.3%	0.2%	1.61	$922.96	$2,756.30
ALL	248 ACTIVE CLIENTS	6,510	30.7%	16.3%	2.0%	10.7%	8.6%	31.6%	27.1%	230.56	$132,573.45	$789,021.72

Figure 4. TOP 40 Demand Batch Clients by Department

NOVEMBER 1987
TOP 40 DISK STORAGE CLIENTS BY MEGABYTE-HOUR OCCUPANCY
FOR INVESTIGATIVE FUNCTIONS

RANK	CLIENT ID	CLIENT NAME	MB-HOURS PERMANENT DATASETS	MB-HOURS TEMPORARY DATASETS	TOTAL MEGABYTE-HOURS	% MONTHLY CLIENT/DEPT. MB-HOURS	ESTIMATED MONTHLY COST	ESTIMATED YEAR-TO-DATE STORAGE COST
1	D0911	SCOTLAND YARD FINGERPRINT FILES- DEPARTMENTAL	108,648		108,648	0.4%	$543.24	$2,170.05
2	X0288	WIMSEY, P. D. B.	86,637		86,637	0.3%	$433.19	$1,834.36
3	E0007	BOND, J.	60,837	561	61,398	0.2%	$306.99	$1,372.25
4	RA277	HOLMES, S.	59,807	19	59,826	0.2%	$299.13	$795.25
5	VC205	VANE, H.	49,250	3,094	52,344	0.2%	$261.72	$1,054.08
6	LB312	POIROT, H.	50,280	909	51,189	0.2%	$255.95	$1,162.52
7	QC478	DALGLIESH, A.	44,467	5,320	49,787	0.2%	$248.94	$888.88
8	WA730	BROWN, F.	46,827	2,956	49,783	0.2%	$248.91	$801.60
9	D0266	MUGSHOT CENTRAL - DEPARTMENTAL	48,908		48,908	0.2%	$244.54	$1,111.03
10	I1626	SPADE, S.	41,615	1,620	43,236	0.2%	$216.18	$838.11
11	HN274	WATSON, T.	39,984	2,249	42,233	0.2%	$211.17	$1,018.51
12	FA119	MASON, P.	36,297	2,942	39,239	0.2%	$196.20	$720.29
13	R0319	QUEEN, E.	37,238		37,238	0.2%	$186.19	$672.82
14	TA728	EGG, M.	34,676	1,875	36,551	0.1%	$182.75	$609.97
15	YN141	WOLFE, N.	32,351	3,655	36,006	0.1%	$180.03	$536.19
16	GA385	STREET, D.	29,702	107	29,809	0.1%	$149.05	$873.16
17	TC056	BROWN, E.	22,094	2,237	24,331	0.1%	$121.66	$412.28
18	W1332	LESTRADE, I.	23,287	462	23,749	0.1%	$118.74	$578.49
19	NC352	BROWN, P.	20,941	2,367	23,307	0.1%	$116.54	$347.21
20	GA755	MARPLE, J.	21,515	1,274	22,790	0.1%	$113.95	$574.73
21	UC182	KELLING, S.	21,415	722	22,137	0.1%	$110.69	$412.49
22	P0767	ROCKFORD, J.	19,784	1,289	21,073	0.1%	$105.36	$375.68
23	K1534	BITTERSOHN, M.	18,979	1,344	20,323	0.1%	$101.62	$274.74
24	LB656	FRIDAY, J.	19,653	566	20,219	0.1%	$101.09	$450.81
TOTAL FOR TOP 24 CLIENTS/DEPARTMENT LIBRARIES			975,194	35,568	1,010,762	4.1%	$5,053.81	$19,885.50
TOTAL FOR ALL 301 CLIENTS/DEPARTMENT LIBRARIES W/ DATASETS			1,455,298	70,025	1,525,323	6.2%	$7,626.61	$30,762.29

Figure 5. TOP 40 Disk Storage Occupancy Clients by Department

NOVEMBER 1987
TOP 40 CLIENTS OF D.P. SERVICES FOR SAVOY DIVISION
8AM - 5PM

RANK	ID	CLIENT NAME	TSO ESTIMATED MONTHLY COST	DEMAND BATCH ESTIMATED MONTHLY COST	DISK STORAGE ESTIMATED MONTHLY COST	TOTAL ESTIMATED MONTHLY COST	TOTAL ESTIMATED YEAR-TO-DATE COST
1	G6987	PLAZA-TORO, C.	$75.18	$7,801.74	$26.62	$7,903.55	$7,903.55
2	I7682	TOLLOLER, E.	$158.52	$5,700.31	$58.18	$5,917.01	$26,711.83
3	PI749	GAMA, A.	$392.89	$4,759.84	$403.46	$5,556.19	$9,840.06
4	S6909	SANGAZURE, A.	$31.98	$5,105.79	$14.52	$5,152.29	$21,662.27
5	M5220	BAH, P.	$77.15	$4,926.29	$15.79	$5,019.23	$8,266.86
6	UL588	FITZBATTLEAXE, C.	$80.45	$4,767.99	$40.64	$4,889.09	$58,141.33
7	GD628	JELLICOE, J.	$49.75	$3,115.14	$29.46	$3,194.35	$10,447.11
8	R4489	GOODHEART, A.	$57.23	$2,662.64	$61.65	$2,781.52	$8,215.22
9	HMSP9	RACKSTRAW, R.	$181.05	$2,480.26	$38.93	$2,700.25	$5,352.07
10	M5322	YUM, Y.	$8.49	$2,216.76	$8.05	$2,233.30	$2,325.33
11	TBJ78	JUDGE, A. G.	$729.48	$1,294.37	$139.60	$2,163.45	$12,126.26
12	Y8951	SHADBOLT, W.	$1,017.87	$245.56	$360.87	$1,624.30	$5,661.87
13	HMSP7	CRIPPS, B.	$264.84	$1,291.50	$65.75	$1,622.09	$6,588.10
14	Y1234	CARRUTHERS, K.	$1,276.51	$0.00	$175.90	$1,452.41	$7,588.56
15	UL534	ZARA, P.	$322.28	$761.52	$71.70	$1,155.50	$9,639.23
16	POP96	KING, A. P.	$234.02	$793.40	$16.78	$1,044.20	$2,777.84
17	HMSP1	PORTER, S. J.	$54.93	$925.04	$13.93	$993.90	$9,055.94
18	P8468	BUNTHORNE, R.	$895.15	$15.77	$70.94	$981.86	$5,718.32
19	S1626	WELLS, J. W.	$124.62	$581.06	$226.89	$932.57	$3,881.91
20	R6671	DAUNTLESS, R.	$874.90	$6.37	$33.62	$914.89	$2,226.74
21	G7312	PALMIERI, M.	$593.30	$0.07	$279.03	$872.39	$5,229.93
22	M1702	KO, K.	$44.24	$757.32	$11.60	$813.16	$1,637.78
23	GD896	CORCORAN, E.	$345.03	$402.97	$57.42	$805.42	$3,222.09
24	Y0099	MERYLL, L.	$530.68	$124.87	$93.59	$749.13	$11,452.75
25	S1332	POINTDEXTRE, M.	$84.32	$561.09	$97.09	$742.50	$15,786.39
26	PI141	HILDEBRAND, H.	$494.73	$80.02	$138.97	$713.71	$5,319.60
27	GD656	DUMMKOPF, E.	$464.41	$141.31	$86.44	$692.16	$4,439.15
28	R8895	MAYBUD, R.	$618.98	$15.47	$53.84	$688.29	$3,831.89
29	M4268	SING, P.	$569.86	$0.00	$75.68	$645.54	$3,739.78
30	P8852	GROSVENOR, A.	$485.91	$0.00	$99.30	$585.20	$3,958.00
31	GD757	VON KRAKENFELDT, B.	$515.23	$24.67	$40.28	$580.18	$4,115.78
32	G1016	DEL BOLERO, A.	$201.68	$242.36	$64.97	$509.00	$3,909.28
33	Y5120	POINT, J.	$457.72	$0.00	$7.23	$464.95	$464.95
34	M4274	POO, N.	$221.05	$12.36	$222.00	$455.40	$4,176.61
35	R0211	MURGATROYD, R.	$0.00	$0.00	$410.22	$410.22	$3,479.85
36	S7061	PARTLETT, C.	$58.03	$349.49	$1.32	$408.84	$498.97
37	POP11	STANLEY, M. G.	$26.29	$344.99	$35.10	$406.38	$8,581.14
38	Y3277	MAYNARD, E.	$78.88	$28.02	$295.76	$402.67	$2,513.35
TOTAL NETWORK USAGE FOR 321 USERS			$16,566.78	$54,088.04	$6,732.55	$77,387.37	$167,450.38

Figure 6. TOP 40 Clients of D.P. Services by Department

CONCLUSIONS

Based on the response within our organization to the reporting techniques discussed here, I believe it is essential to work toward implementing and refining some version in any organization where the IS function is a service function for the overall business. Computer resources consume an impressive portion of the overall corporate dollar, and non-IS management generally do not understand their units' use of these resources well enough to manage it for maximum effectiveness. It is up to the technical wizards in IS to come at least halfway to help client department management to manage their IS resource usage. As direct consumption of these resources by client departments continues to grow, IS cannot manage them alone. Better management by IS requires client department participation.

Through better communication and cooperation between IS and client department management, this type of reporting can improve IS' standing in the corporation. At the same time, managing the installed capacity will be made easier through sharing the load. Both IS and the rest of the corporation will be the ultimate winners.

REFERENCES

[1] CMS, IMS, SPF, and TSO are trademarks of IBM.

[2] MICS is a trademark of Morino Associates.

[3] RAMIS is a trademark of Online Software.

[4] SAS is a trademark of SAS Institute, Inc.

Management Reporting: An Approach To Availability and Service Reporting

Michael Marcus
Freddie Mac

ABSTRACT

Management needs and wants timely information about service and availability of the online systems. This paper presents a definition of availability, traces the history of the different reporting methods developed over the years, and reports on a recently developed availability management reporting system. For any installation not currently doing this type of reporting, the information presented can be used as a starting point for developing a set of management reports.

INTRODUCTION

One of the problems of providing reports to management is reporting technical information in a form that is easy to understand and does not overwhelm the reader. Sometimes trial and error prevail in the development of reporting systems. The following availability and service reporting system has evolved in part by trial and error and in part by a need for better reporting. The current set of reports and graphs have been used at all levels of management. What this paper is attempting to do is give some examples of what has been done in this area of reporting to provide a starting point for developing a set of reports and graphs for use in any installation.

When availability and service are at "acceptable" levels, management has not always demanded reports about service indicators. Until there are many interruptions in availability or service becomes degraded, management does not need this information. This is especially true of organizations that are "new" to online data processing. When this happens, management has a need to know what is happening and why. To be able to provide this information, data from multiple sources needs to be collected and maintained.

AVAILABILTIY

Before producing a report on availability, a concise definition of availability is needed. Our definition for availability is the measure of time a given service is available to end users within established service hours. If a given computer service is scheduled for 12 hours and is in operation for 14 hours, it is only considered for tracking purposes to be available for 12 hours. There is no bonus given if an outage occurs and service is extended to compensate for the outage. If an outage is scheduled during normal service hours, there is no penalty provided service is restored within the scheduled time. For single image services provided by multiple servers (TSO from multiple machines or a split CICS application), a dilemma exists on how to report an outage in part of the system. Service is still being provided, but all users may not be able to access the service. The outage can be averaged between the services or counted as a full outage. Management has chosen to average these outages.

In order to reflect the view of availability seen by the end users, network data is included in the reports. This is indicated by reducing the percentage reported for each service location(see report, figure 2). Network outages can pose some problems. If a location is serviced by more than one line but is reported as a single service location, the percentage of availability is calculated by dividing the total available hours for all nodes into the total scheduled hours for all the nodes. This method accounts for outages to only a portion of the service location.

AVAILABILITY DATA COLLECTION

Prior to this year, the availability data was manually collected by the computer operators, summarized by the operations shift supervisor, and reported only to the data center management. For upper management, the data was summarized and reported with other management summaries. A computer-assisted method for collecting availability data was developed. Commercially available systems to collect and report information about system availability required some manual adjustments, a decision was made to develop a system rather than use one of the existing products since it could be developed rather quickly. The existing data collection products can only report when the service begins and ends on the computer system, and may not reflect when the service is actually available. As an example, if CICS abends at 10:27 and is restarted at 10:28, traditional data collections methods will report a 1 minute outage. In fact, CICS may have taken 2 minutes to dump after the abend message was received and 5 minutes to restart to the point of restoration of service for a total outage of .1 hours or .15 percent for a service scheduled 68 hours per week. While being a trivial amount, for an unstable system experiencing many of these outages, this could be of note to management.

The availability data entry system uses SAS® to build a daily schedule for the online services to be tracked. Each service is then displayed with blank fields for the computer operators to enter the actual start and stop times and JES job number(see figure 1). Additional fields are available to enter outage reasons should an unscheduled outage occur. This is used to provide statistics about the types of outages that occur. Five sets of these fields exist for the possibility of 4 unscheduled outages on a single day. For systems with scheduled outages, multiple records are required for each set of service times. After the data is entered the hours and percentage available are calculated and the availability data base is updated with the current day's entries.

The network availability data has not been as easy to collect and maintain as the host availability. Automated methods were tried for collecting outage data. The use of NPDA alerts was attempted. There were problems in determining outage start and end times from the available data. All outages are recorded by the system, but depending on when and how service is restored it was not always possible to match the two records. The outage data is collected outside of the mainframe using a PC, uploaded and summarized. Because some network outages may recover themselves quickly, sometimes without the users being aware other than slow response, it was decided to report only outages that are longer than 3 minutes. Outages are applied to the standard service hours using the methodology previously described. The data is then merged with the host availability data to produce a consolidated number for each node reported.

```
                Edit SAS data set: AVAIL.DATA              Screen  1
Command ===                                                Obs  1792

 DATE:  07DEC87   JOB NAME: CICSPROD    SYSID: MVSS   USERID: MARCUS
 SCHEDULED UP TIME:   08:00    SCHEDULED DOWN TIME:  20:00
 SCHEDULED HOURS: 12:00  ACTUAL HOURS: ____  PERCENT AVAILABLE: ____

    ACTUAL   ACTUAL   JES              REASON
    START    STOP     JOB      REASON  FOR
    TIME     TIME     NUMBER   CODE    OUTAGE
1:  07:12    15:22    0176     SOFT    ABEND U0845 - IMS PROBLEM
2:  15:30    20:08    2468     ____    ____________________________
3:  _____    _____    ____     ____    ____________________________
4:  _____    _____    ____     ____    ____________________________
5:  _____    _____    ____     ____    ____________________________

DELETE at Command to delete record.
Use seperate entries for unscheduled systems.
PF3 - END   PF9 - Add new record   PF8 - Next   PF7 - Previous
```

Figure 1. Example of On-line Data Entry Screen

SERVICE DATA

The consolidated report of service and availability includes data on response time. Response time data has always been misunderstood. There is host response and network response, and depending on how the measurements are taken and the software products used, a consistent response picture may not be accurately reported. This is not a problem as long as everyone reading the report is aware of the measurement. CICS host response begins when the transaction is started within CICS. Network time or queueing within CICS prior to transaction start is not counted. With definite response turned off, CICS response time ends after the last message is sent and the transaction terminates. If definite response is turned on, the CICS transaction waits for the terminal / controller to acknowledge the receipt of the screen. This may add a significant volume of data transmitted across the lines, thereby increasing response time. In one instance when definite response was accidentally turned on, response increased by a half second. Some measurement products require that definite response be used to record host response.

RTM, a hardware feature that collects response data, is available on IBM 3274 controllers. Using the NLDM or NETVIEW software, RTM collects statistics for defined services. The records are written to SMF as type 39's. The data is summarized by the service locations defined for reporting. One of the problems using the RTM data is that any devices not connected to a controller with the RTM feature (Tandem, Wang, LANs, etc) will not have any response time data. If this is a significant portion of the network, other alternative methods of collecting the data may be necessary.

THE REPORTS

The consolidated report (figure 2) shows the breakdown by service and service location. This report has been sent to the end-users. The columns (A, B, C, and D) show the availability of the service, the average network response time, the percent of network transactions under 3 seconds, the percent of network transactions under 5 seconds response time. The report is produced on the mainframe. If any adjustments are required,they are made prior to uploading to the electronic mail system for distribution.

On a monthly basis the data is combined with other host and non-host service indicators to produce a set of graphs(figure 3) for upper management. These include the service indicators described above and others such as word processing and local area networks. Much of data collection has not been automated, thereby requiring considerable effort to collect and tabulate the data. These graphs have been extremely well received by management.

THE REPORT HISTORY

Much of the data in the report (figure 2) evolved from a number of internal reports. A consolidated report was needed for management.

One of the first reports developed detailed a profile of CICS internal(host) response time. The report was sorted and summarized in various ways. The report was not effective because it presented too much data for management and could be misleading. The report showed that host response time was good, even though there were numerous complaints about response. A summary report was developed to show a response breakdown for the different service locations. This report presented a more concise picture about internal response time. In the CICS response breakdown report, data was summarized for each CICS service by service location. The report detailed the average response, percent of transactions completing within 3 seconds, the number of transactions by response limit.

The first version of the network response summary report directly summarized the SMF type 39 (RTM) records. The volume of data that was required made producing the report quite time consuming on a weekly and monthly basis. Software was used to summarize the data on daily basis so that reporting could be on a more timely basis. The RTM data is reported in the same manner as in the CICS report described above.

SUMMARY

Online computer services are a primary business function. There needs to be tracking of the service given to different users of the services provided. This system may not provide the data needed by all installations, but those that are not compiling the information or producing reports for management on some of these indicators may not be able to present a picture of the service given to the users. Graphics is a preferred way of showing the impact of the numbers to management. With a little time and effort, a meaningful set of reports and graphs can be created.

COMPUTER SERVICES AVAILABILITY FOR THE WEEK ENDING DECEMBER 6, 1987

SERVICE BY LOCATION	CICSPROD				CICSCORP				CICSDATA			
	A	B	C	D	A	B	C	D	A	B	C	D
NORTHEAST	96.4% /	1.3 ,	97.0% /	99.2%	96.5% /	3.3 ,	89.8% /	93.8%	96.5% /	3.0 ,	84.2% /	87.7%
SOUTHEAST	98.1% /	1.5 ,	96.7% /	99.1%	98.2% /	4.3 ,	84.9% /	90.2%	98.2% /	1.9 ,	77.5% /	93.0%
NORTH CENTRAL	96.4% /	1.5 ,	93.5% /	98.2%	96.5% /	1.8 ,	97.1% /	97.9%	96.5% /	2.3 ,	88.7% /	92.0%
SOUTHWEST	96.5% /	1.7 ,	94.6% /	98.5%			----		96.6% / *	4.2 ,	70.4% /	77.8%
WESTERN	99.8% /	1.5 ,	96.0% /	98.7%	100.0% /	2.7 ,	88.2% /	92.7%	100.0% /	2.2 ,	90.6% /	94.3%
RESTON # 1												
1ST FLOOR	99.8% /	1.5 ,	95.6% /	98.5%			----		100.0% /	1.9 ,	89.8% /	93.7%
2ND FLOOR	99.8% /	2.0 ,	87.3% /	96.2%	100.0% / *	1.6 ,	75.0% /	87.5%			----	
3RD FLOOR	99.8% /	2.5 ,	78.0% /	92.5%	100.0% / *	2.3 ,	75.0% /	92.3%	100.0% /	2.4 ,	87.9% /	95.0%
RESTON # 2			----				----				----	
RESTON # 3			----				----				----	
RESTON # 4	99.8% /	1.3 ,	96.3% /	98.5%			----				----	
SUNSET PLAZA	99.8% /*	1.5 ,	95.4% /	97.7%	100.0% /	1.0 ,	98.6% /	99.5%			----	
HOME OFFICE												
1-C			----				----				----	
STOREFRONT	96.1% /*	1.7 ,	91.7% /	100.0%			----				----	
3RD FLOOR	96.1% /	2.0 ,	87.9% /	97.7%	96.5% / *	1.2 ,	91.4% /	97.1%			----	
4TH FLOOR	96.1% /*	1.8 ,	91.9% /	96.7%	96.2% / *	2.6 ,	58.3% /	95.8%			----	
5TH FLOOR	99.2% /	2.5 ,	77.9% /	93.1%			----				----	
SUMMARY	98.0% /	1.7 ,	91.5% /	97.4%	98.2% /	1.3 ,	97.1% /	98.6%	98.3% /	2.3 ,	88.0% /	93.2%

A - AVAILABILITY = ACTUAL SERVICE HOURS / SCHEDULED SERVICE HOURS (SERVICE OBJECTIVE IS 98.5%)
B - AVERAGE RESPONSE TIME(NETWORK) = AVERAGE OF ALL TRANSACTIONS
C - RESPONSE TIME(NETWORK) = PERCENTAGE OF TRANSACTIONS PROCESSED WITHIN 3 SECONDS (SERVICE OBJECTIVE IS 90%)
D - RESPONSE TIME(NETWORK) = PERCENTAGE OF TRANSACTIONS PROCESSED WITHIN 5 SECONDS (SERVICE OBJECTIVE IS 95%)
--- = NO ACTIVITY FOR THIS PERIOD OR COMPLETE DATA NOT AVAILABLE
* - THE UTILIZATION OF THIS GROUP REPRESENTS LESS THAN 1% OF THE NUMBER OF TRANSACTIONS

Figure 2. Consilidated Availability and Service Report

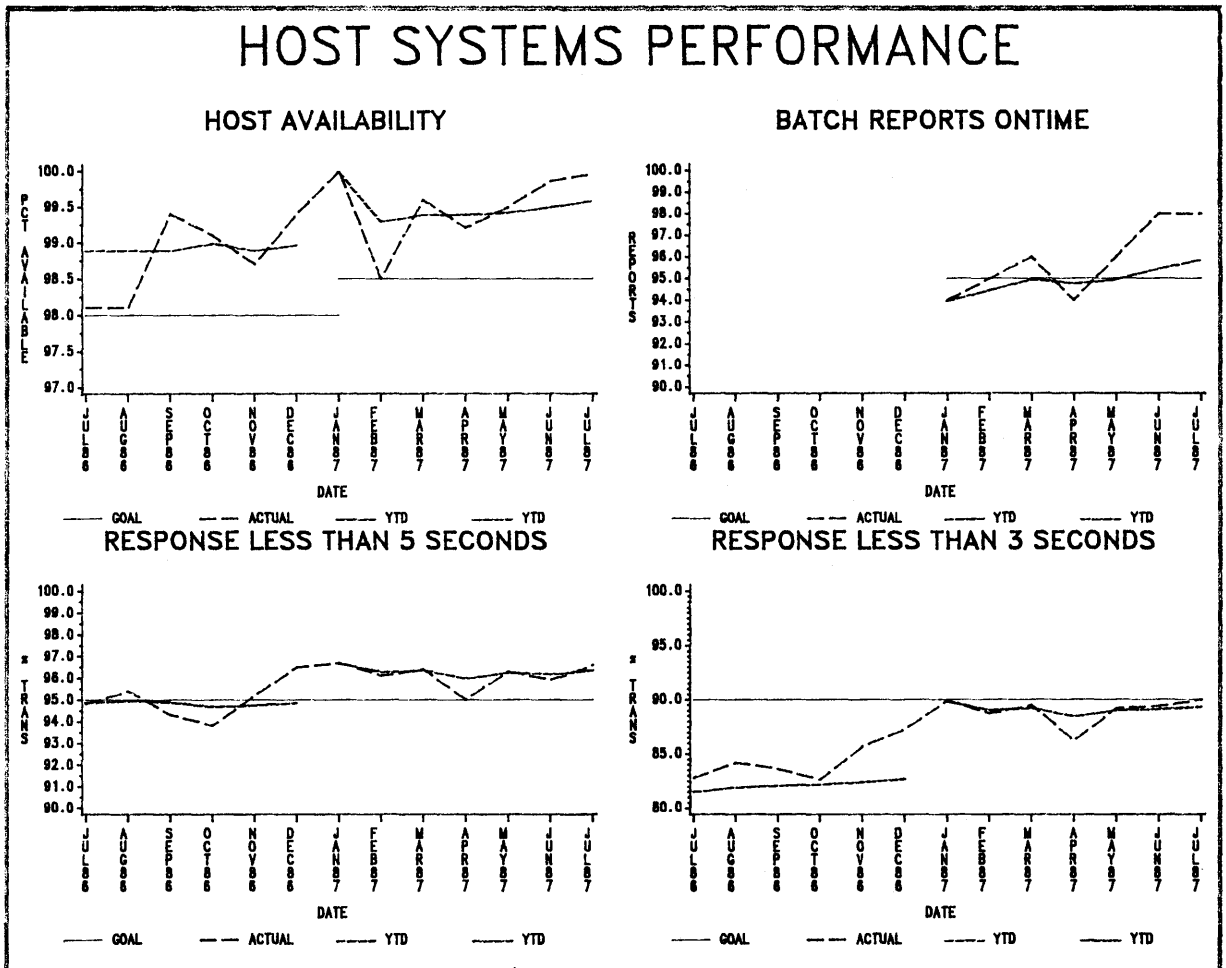


Figure 3. Graphs

QUALITY IMPROVEMENT IN CAPACITY MANAGEMENT

Brian P. Wessner
Amdahl Corporation

ABSTRACT

Only a few short years ago the practitioner of Capacity Management was considered more of an artist than a person performing a rigorous analysis of future data processing resource requirements. As computer hardware has increased in power, and the resulting workloads becoming orders of magnitude more complex, the "art" of Capacity Management has been transformed into much more of a science. As the complexity of the analysis increased, the quality of that analysis has become more critical than ever. What were accepted "rounding" inaccuracies can now become significant shortfalls in system capacity. With today's business so dependent on automated systems for their lifeblood, shortfalls in system capacity can have disastrous results. This paper describes an integrated four-stage approach to the science of capacity management. It then proposes a method to enhance the quality, and hence the effectiveness and usefulness, of the process.

HERE'S THE SYMPTOMS

Have you ever had to ask yourself any of these questions:

* What happened to response times when we put up the new application? I thought our capacity forecast said we could handle it OK!

* We have a DASD management program, but how did we run out of available space so soon?

* We estimated the size of the new system with an analytic model, but why didn't we even come close to what we needed?

* Why have our workload levels been all over the board?

These and dozens of other questions like them have been asked by and of those having established capacity management programs. Having these questions asked by you, the manager of the capacity management program, sheds some doubt on the effectiveness of the program you worked so hard to get started. When these questions are asked OF you, they can be extremely embarrassing or potentially career-limiting if posed by upper levels of management.

WHAT'S THE PROBLEM?

You've had a good capacity management program going for some time now. The useful life of processors and DASD have been extended more than once; scheduled upgrades have occurred when you predicted they would; effective workload management has improved user service levels. How could changes in resource requirements be so grossly underestimated that system performance "went down the tubes"? Not only were business systems affected, but also your department's hard earned reputation for providing quality service.

Let's look at a different aspect of the process not normally included in most capacity management programs: the QUALITY of the capacity management program itself.

A TYPICAL CAPACITY MANAGEMENT PROGRAM

Any capacity management program is designed to take the existing environment, introduce anticipated growth projections, and forecast the timing and the amount of capacity upgrades. Input to the process comes from a variety of sources. Each data element describes needs at different organizational levels. This data must then be integrated into a single resource forecast. The bulk of the data translation and resource forecasting comes from within the capacity management process.

Let's look at a typical capacity management program and consider the major steps in the method, the inputs, the major outputs, and the decision process for handling future capacity requirements. The four stages are summarized in Figure 1.

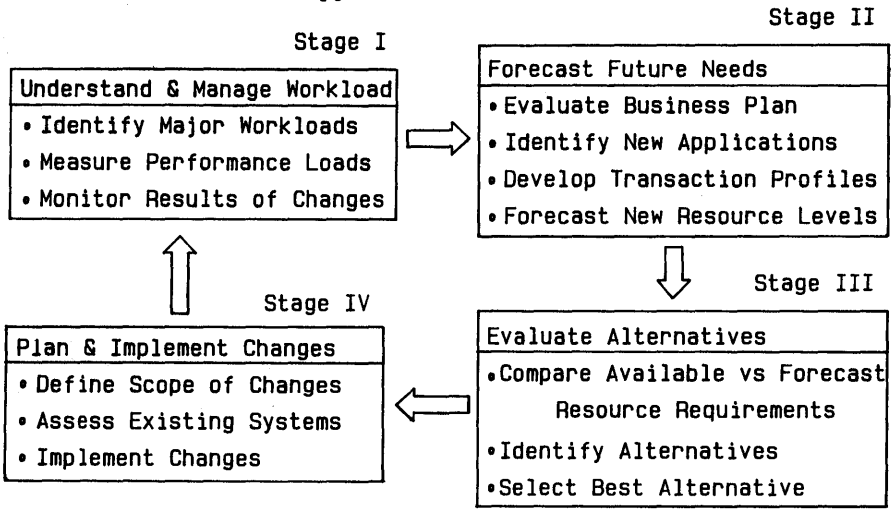


Figure 1. Four-Stage Capacity Management Process

The first stage in the capacity management process is to UNDERSTAND AND MANAGE THE EXISTING WORKLOAD. This objective can be achieved through:

(1) identifying major workloads to be evaluated. This should include on-line applications environments (CICS, IMS, etc.), typical on-line support systems (TSO, etc.), and batch or other background workloads that do not have real-time resource needs.

(2) measuring the performance provided by the overall system and the resource consumption levels required to provide that level of performance. These values can be developed using system-wide monitors such as RMF or VMMAP, job-oriented reporting systems such as JARS, and network monitors.

(3) Once the first two tasks are completed, monitoring and specific tuning actions can ensure continued satisfactory performance levels. Tasks (2) and (3) will then become the ONGOING FIRST STAGE after the capacity management program is in place.

The second stage of this capacity management process is directed at FORECASTING FUTURE NEEDS to be supported by the overall system. This stage includes:

(1) evaluating the current business plan to identify new or significantly changed business strategies.

(2) translating the business plan into requirements for new or enhanced applications systems.

(3) defining anticipated transactions, and their profiles for each of the new or existing applications.

(4) forecasting adequate resources to process the anticipated workload and still continue to meet service level requirements.

The third stage focuses on the EVALUATION OF ALTERNATIVES TO MEET THE FUTURE REQUIREMENTS. The forecasts developed in stage 2 are used as the primary input to this stage of the process. Based on those forecasts, the process then addresses:

(1) predicting future performance of the existing resource against the forecast from stage 2, and

(2) evaluating alternate configurations if existing capacity cannot handle forecasted workload levels.

The fourth and final stage of the capacity management program addresses PLANNING FOR AND IMPLEMENTING CHANGES TO EXISTING SYSTEM RESOURCES. Specific resources have been identified in stage 3. The next stage is to plan for the implementation of those resources. Potential changes are not limited to additions to the current system configuration; changes may include hardware reconfiguration, new or enhanced procedures, or tuning of available system components.

Once changes have been made to the current environment, return to the first stage of the process and continue to monitor overall system performance.

WHAT IS "QUALITY" ALL ABOUT?

Before we can point to those things that result in a "quality" capacity management program, let's look at some of the characteristics of any quality improvement program.

First we need to define "quality." We all know it when we see it, but "it" is typically a subjective impression or opinion: the new car that doesn't require continual trips to the dealer; the craftsman who takes deep pride and satisfaction in what he makes (many times by hand); or the capacity manager who is determined to develop accurate, timely forecasts of future computing resources. The bottom line definition of quality is "conformance to requirements." When someone is asked to "do it right the first time," the "it" is the requirement; in our case, the capacity management process.

Doing it right the first time defines an explicit standard of performance. DO IT RIGHT ALL THE TIME - not 99 percent or 90 percent, but all the time. Let's say that an airline mechanic attempts to repair nine out of ten planes correctly the first time. Would you want to be a passenger on that tenth plane?

How do you get things right the first time? Do you build a complex system of checks and balances to catch errors and then attempt to correct each one? With that approach the error has already occurred and you're adopting a "damage control" approach to correcting the problem; or you have to repeat the process again, adding a new control "on the fly" to prevent the error from happening a second time. Why not design the the system itself to PREVENT errors from happening? By specifying the system up front, before actually using it, you can define exactly what requirements are to be met. You can then identify where errors are most likely to occur and build in appropriate actions to prevent those errors from happening.

How do you measure quality? Should you devise one or more measures or yardsticks of quality performance? What about a "quality index" based on a mathematical formula? These approaches do not get to the bottom line: lack of quality costs your company MONEY. The money may be spent to modify the process, to buy and store excess inventories, to rework goods already manufactured, or worse yet to account for lost revenue from a dissatisfied customer.

How to Improve the Quality of Your Capacity Management Program

A capacity management program, like any other process-oriented activity, takes specific inputs, acts on those inputs through a series of processes, and produces outputs. The output, a product or service, is provided to someone. That someone can be called the "customer." To produce a satisfactory output you must understand the expectations, expressed in terms of requirements, that are in the mind of the customer. Viewing DP as a service-oriented organization that supports business goals and objectives, we must consider the ultimate customer to be the business manager that depends on those services.

What is the requirement, the "it" to be met? For most capacity management programs the requirement can be defined as:

> Ensure that sufficient computing resources, of the type necessary, are provided in a cost-effective manner to support successful business operations; the resources are to be available when they are needed and they must perform to agreed-upon standards.

Let's go back to the four-stage capacity management process and see how the quality of the process can be improved.

In the first stage we identify the major components of the overall workload. But how do we define "major"? Any application that assists a critical business function should be considered a major application. Manufacturing

companies rely heavily on order processing, materials management, inventory control, and order distribution systems. Technology-based organizations depend on design and engineering applications. Financial institutions conduct business through automated teller machine (ATM) networks. Other industries can also identify equally appropriate major systems that contribute to the success of the business.

Performance expectations are then used to determine how the current environment "measures up" against requirements for each major system. Service level agreements may be used to define user expectations. These requirements should be expressed in business terms understood by and agreed to by both the business unit and the DP organization. For example, an insurance company may measure business success by the number of new policies that are processed in a given time period. The performance of a policy processing system should therefore be measured in the same terms. DP then translates that need into specific resource consumption levels per unit of work, such as an online transaction.

The second stage of the capacity management process focuses on forecasting future needs. This stage combines inputs from a variety of sources into a single measure of anticipated requirements. It is important to include high-level forecasts, such as those found in the business plan, in the forecasting process.

Many times DP does not have access to the business plan to help in the forecasting process. Even if the plan is available, it may not describe projected sales growth, the timing of new products or service offerings, or the acquisition of additional business units. To reduce these uncertainties, DP should work with the major business units to define a set of possible business scenarios. Each scenario would be based on changes in key business growth assumptions. DP would then develop a range of alternatives based on each scenario. General management would assign a probability to each business scenario to guide the evaluation of potential capacity requirements.

Anticipated transaction volumes, as well as their profiles, should be tested for "reasonableness" against historical growth projections. If last year's business plan projected a 10 percent increase in orders, did the order entry system actually process 10 percent more transactions than the year before? What about the year before that? Compare the actual versus projected volumes to establish a trend in the projections published in the business plan. If projections are consistently higher or lower than the measured volumes, future forecasts should be adjusted accordingly. Processing done by today's system probably differs from when the system was originally implemented. Yesterday's queries may have become today's updates that consume significant computing resources. Only by tracking transaction profiles, and comparing those profiles to the planned business strategy, can consistent resource forecasts be made for the applications portfolio.

Resource forecasts are directly affected by the confidence placed in the business and application projections used to develop those forecasts. Forecasts developed in a uncertain planning cycle can be identified as "best case," requiring the lowest increase in resources; the "worst" case, associated with the highest growth scenario; and the "most likely" case with the highest probability of occuring. These scenarios then form a range of forecasts based on the business climate, not the computing environment. Business management can then direct the actions of DP based on the strategy it selects to run the business.

As we enter the third stage of the process the business environment has been defined and the resource requirements have been established. The next questions to consider are: "Can the current configuration handle the forecasted requirements? Will service levels continue to be met?" If the answers are yes to both questions then nothing more is necessary during this stage. If additional resources are required, what kinds of resources are they? For example, increased I/O resource usage can be accommodated through:

* adding new storage strings,
* caching frequently read program or data files,
* installing electronic data storage devices,
* reconfiguring of existing data storage resources,
* moving data to lesser used storage resources, or
* implementing a data archival program.

Each of these actions has its own advantages and disadvantages. Each action can address the I/O resource issue. The job of the capacity management process is to identify these potential actions and combine one or more of these actions into alternatives that can be evaluated by upper management.

What criteria does the general manager use to evaluate any set of alternatives? The manager considers the COSTS, the potential BENEFITS, and the RISKS associated with selecting each alternative. Recommendations (outputs) provided by the capacity management program should be presented in those terms; the process itself must then be structured to evaluate alternatives based on cost, benefits, and risk. Alternatives presented for management review may not be acted upon. The decision to bypass an alternative should be a conscious one made based on information that is presented in terms which can be understood by the business decision-maker.

Once the most desirable alternative has been identified the fourth stage, that of implementing the alternative, can begin. As with any change having a significant business impact, an action plan should be developed. The plan should identify resource additions, define organizational responsibilities for specific change activities, coordinate activities with the overall system change plan, and track progress against the plan. To plan effectively for and implement system configuration changes the process should address:

(1) identifying specific hardware and software components or procedures affected by the change;

(2) timing both the overall change as well as individual components of the change. For example, no change should be attempted during peak processing times such as year-end closings. A major change can be broken into smaller segments when possible;

(3) continuing satisfactory system performance while the change is being made. If performance may degrade while the change is being made, the duration and extent of performance degradation should be considered before going ahead with the change. If the impact of the change is too great, then other alternatives should be sought;

(4) minimizing the risks associated with the planned changes. Risks may relate to availability, performance, accuracy and timeliness of processing results, access to resources, or opportunity costs to the business if automated systems cannot be used while the change is in progress; and

(5) creating FALL-BACK PLANS to address each change. Each fall back plan should define the extent of the planned change, parameters measuring the success of the change, guidelines to decide if the change

should be "backed off," and the procedures to perform the fall back. The fall-back procedure should be tested and approved prior to actually implementing the change.

Resource modification attempts to eliminate resource shortfalls or reduce contention for available resources. Each change that is made carries with it specific expectations for improving resource capacity. Improvements following the actual change must then be compared to the expected change. But what criteria should be used to measure the success of the change, and how are these criteria applied by the capacity management process?

Before even starting this validation process, one important question must be answered: "What will happen if the change does not meet the stated requirement?" If no action will be taken then go no further. You are wasting your time with a validation process that will never improve the quality of your capacity management decisions. The answer to the question should be: "We will find the reason why the process (or data used within the process) did not provide an accurate forecast. We will update the process (or its data) to ensure that this error does not occur again."

How do you design this validation process? Start by identifying key factors that determine the success, or the conformance, to the requirement that you are attempting to meet. Then establish an acceptable measure of achieving that objective. For example, if an increase in processor capacity was expected to decrease response time by 15 percent and the actual improvement was only 5 percent, then the forecasting process or the input to the process was in error. Now locate the source of the error by reviewing the steps of the process that were used in this decision. How does this evaluation differ from previous successful evaluations? Was the data different? Was this the first evaluation since a major shift in business requirements or did the business shift occur AFTER we started the analysis? Did we recently install a new data collection or analysis tool?

Once the error has been found, identify corrective actions. These actions should include:

(1) describing the changes to be made,

(2) testing the proposed changes,

(3) deciding who will implement the changes,

(4) documenting the changes, and

(5) training those involved in the updated process.

Just as when making changes to computing resources, changes to the capacity management process require careful consideration and planning in order to meet the requirements of the process.

OTHER ELEMENTS IN QUALITY IMPROVEMENT

Up until now we have focused on improving the quality of the process itself. Let's shift our discussion to areas beyond the actual process that can contribute to an overall improvement in quality.

Quality of the process can be expected to improve as we remove obstacles that prevent people from doing the job right the first time. Typical obstacles can be removed by:

(1) providing adequate facilities, such as computer time for modeling exercises,

(2) installing tools needed to enhance or streamline the analysis process, such as financial evaluation spreadsheets,

(3) developing a training in sound capacity management techniques, project management, change control, and negotiation techniques,

(4) defining capacity management policies and procedures to ensure the consistency of the process, and

(5) enlisting management support and commitment to the success of the capacity management program

Each of these items can make the capacity management process more effective for the business it supports. Taken together, they can ensure the development of a QUALITY capacity management program.

SUMMARY

A quality capacity management program can result in tangible benefits to the bottom line of the business organization. In addition, DP's reputation can be strengthened immeasurably through timely, accurate, and business oriented management of future computing resources. The steps that were discussed in this paper form the framework for the development of a quality capacity management program that will benefit the business organization for years to come.

REFERENCES

[1] Amdahl Corporation, Effective Capacity Management Seminar, copyright 1986.

[2] Crosby, P. B., Quality is Free: The Art of Making Quality Certain, Mentor Books, 1980.

[3] Crosby, P. B., Quality Improvement Through Defect Prevention, Philip Crosby Associates, Inc., 1985.

Performance/Capacity Data Base For Studies of Long-Term Trends

Hulan E. Jack, Jr. [1]
Metropolitan Transportation Authority, NY

Kim Goldenberg
State of New Jersey, Dept. Of Treasury, OTIS

Roger Williams
System/Software Engineering

ABSTRACT

There is a constant need for studies to identify, confirm and track long-term trends and problems, and various oscillatory behaviors such as over hours of day, days of week or month, etc. Such studies usually need to be done quickly. As such, the data for them must be conveniently stored. This paper describes a special performance and capacity data base created to maintain appropriate performance and resource usage data on-line for periods covering long time spans, such as up to one or two years at the MTA. It consists of hourly averages for each day in a collection of selected critical performance and resource usage parameters. Samples of quarterly reports generated from it are included.

1. INTRODUCTION

There is a constant need for ad-hoc studies and regular reporting over long time spans, such as months, quarters, or a year or more, that identify, confirm and track long term trends and problems and various oscillatory behaviors such as over hours of day, days of week or month, etc. For this purpose, it is desirable to maintain an on-line data base that contains a collection of critical data for time spans of at least a year, perhaps two years. The rationale for this type of database is developed in an earlier article by H. E. Jack [1]. The original raw SMF data, usually archived to tape, is too voluminous and cumbersome and too inconvenient to use for these purposes. Commercial performance data base systems such as Merrill Consultants' Merrill's Expanded Guide (MXG) or Morino Associates' MVS Integrated Control System (MICS) convert the SMF and RMF data into more convenient to use SAS[2] datasets. Additionally, they contain a collection of packaged reports that are good for general reporting and some troubleshooting, but they are still too voluminous and cumbersome for this use. But, their collection collection of extensive, detailed, conveniently detailed data is ideal for very rapid trouble shooting. These data, whether raw SMF or the more convenient SAS conversions, in their entirety would require 4 IBM 3380E volumes for a month of MTA data. Thus, it would be uneconomical to store all this data on-line. For the objective stated here, a collection of condensed data is required. This paper describes one effort at such a data base. Section 2 is an overview which briefly outlines what aspects of the computer system are of principle interest and the time granularity of the data to be collected and stored. Section 3 describes the method of building the data base along with the selection of items chosen and the rationale behind the choices. Section 4 shows and discusses the quarterly reports presently produced from these data. Section 5 assesses the present system and outlines future directions.

2. OVERVIEW

The first problem is to decide what aspects of the computer system are to be characterized. This determines the specific data base items to be collected. The next problem is to determine the level of time granularity for the data, i.e., instantaneously, hourly, by shift, daily, weekly, monthly, quarterly, etc. These two problems will now be briefly addressed. These issues are developed in more detail in Ref. [1].

2.1 THE FEATURES TO BE CHARACTERIZED

For the purposes of this data base, only the most critical aspects of the computer system should be characterized, at not too detailed a level. Most critical is data on the utilization of the CPUs, the memory, and the I/O subsystem. The next level of criticality is the workload responsible for these utilizations.

2.2 WHAT LEVEL OF TIME GRANULARITY

To represent the state of a system, observations of its key features can be recorded instantaneously (the level of the raw data, i.e., 5 minutes), hourly (averages over the hour), by shift (averages over the shift), daily (averages over the day), etc. Averages over a long period, such as shift or day, do not adequately capture the peak periods. For example, over a 10 hour shift the average CPU utilization could be 40%, while it was 95% for about two hours steady (20% of the shift) and only around 26% the remaining eight hours (80% of the shift). This misses a very serious situation. At the other extreme, data for every 5 to 15 minutes can be too detailed. For instance, a CPU utilization of 99% for a 5 minute interval, with adjacent intervals below 70% usually represents little pain. It can be seen from this that the interval chosen must be long enough for the averages to indicate overall conditions, yet short enough to preserve some temporal detail. An hour is chosen as the interval because it is long enough for averages to indicate overall conditions and short enough to retain temporal details. Additionally, it conforms to our cultural division of the day - lunch hour, happy hour, etc.

---

[1]Present address: FISA, City of New York,
111 Eighth Avenue, NY, NY

[2]SAS is a registered trademark of SAS Institue, Inc.
of Cary NC

3. BUILDING THE DATA BASE

This section first describes the method of building the special data base. Then it discusses and describes the specific data retained and the sources from the performance data base and the SMF record types that are the source of the data base files.

3.1 THE METHOD OF BUILDING THE DATA BASE

For convenience and efficiency, the data base was built from the MICS system in use at the MTA. There is no requirement to use MICS; this can be accomplished with other performance data base systems or with raw SMF data and homegrown programs. Presently, this special data base is updated for use in generating quarterly capacity and performance status reports. A MICS Special data base was established to process the SMF tapes for the SMF and RMF components of MICS. It collects information for about 10 days per update. Available DASD limits this data base to five updates. After five of these updates, another job is run to extract information and update the performance and capacity data base, extracting the hourly sums of the selected data base items. The selections are made by way of SAS Keep statements. In the future, this data base update will be incorporated into the daily MICS update job. Figure 1 diagrams the present method of building and updating this data base.

3.2 THE DATA AND ITS SOURCES

Table 1 shows the collection of the SET and KEEP statements used to build the data base. All of the files are similar in construction to the MICS daily files using MICS type file names; this allows for common reporting procedures and is not a requirement. They contain data averaged or summed over each hour.
HARCPU is the CPU activity file derived from SMF Type 70 records. This file contains data on various user types (TSO, Batch, Started Tasks) in the system as measures of the extent of use (multiprogramming level); the number of users in various wait and ready states as a measure of prospective shortage of CPU resources; the busy, wait and up times, the configuration indicator, and on-line/off-line status for each CPU engine as measures of CPU activity and availability, and the number of various I/O interrupt as measures of the interrupt activity of the CPU.

HARCUA is the control unit activity file derived from SMF Type 73 records. This file contains controller activity and utilization data for measures of usage and request queue levels as a measure of performance problems and capacity shortage. The specific data items differ between MVS/370 and MVS/XA.

HARPCA is the physical channel activity file derived from the SMF Type 73 records. This file contains physical channel activity, utilization and service time data for measures of usage, and queue counts as a measure of performance and capacity problems.

SCPLCA and SCPLCU are logical channel (for MVS/370) and logical control unit (for MVS/XA) activity files derived from the SMF Type 73 records. This file contains data on requests deferred, channel/control unit busy and various queueing statistics as a measure of performance and capacity planning.

HARTCA is the total channel activity file derived from the SMF Type 73 records. The file contains data derived from the combination of physical and logical channel/control unit activity.

HARDEV is the device activity file derived from the SMF Type 74 and, from Type 78 records for MVS/XA only. The file contains data on DASD device activity, utilization and service time data as measures of usage; on number of requests pending, and various queueing data as measures of performance and capacity problems.
SCPPAG is the paging and swapping activity file derived from the SMF Type 71 records. The file contains data on paging, swapping, and virtual memory activity.

SCPPGA is the performance group workload file derived from the SMF type 72 records. This file contains data on the usage of CPU and main memory, as well as I/O and swapping activity for each performance group. In order for this file to be effective, it is important to assign each major software component (CICS, database system, etc.) and application group to an individual control performance group for tracking, even though it may not be necessary for performance or service level control of the group involved.

SCPPSD is the page/swap data set activity file derived from SMF Type 75 records. This file contains data on the usage of each page and swap data set including percent of use and I/O activity.

BATJOB is the job information file derived from the SMF Type 30, subtype 5 and Types 6 and 26 records. The file contains data on input queue time, input job classes and spooling and printing data.

BATPGM, BAT_ST, and BAT_TS are step oriented data for batch jobs and started task and TSO user data files derived from the SMF Type 30 subtype 4 records. The file contains data on TCB and SRB CPU times, I/O counts for DASD, Tape and Unit Record devices, virtual and real memory usage, and paging and swapping data.

4. THE QUARTERLY REPORTS

Figures 2 through 10 show examples of the quarterly reports. The first five, Figures 2 to 6, make up the performance section. The next three, Figures 7 to 9, are the capacity section. Figure 10 makes up the workload section. All graphs except Figure 2 are for Prime Time hours (Monday to Friday, 0800 to 1800, excluding holidays) because this is the most active and most critical period; Figure 2 is for all 24 hours over the whole reporting period. There is heavy use of on-line applications using CICS and IDMS database system, heavy interactive (TSO) activity, and moderately heavy demand batch activity, all running in competition with each other on each of three systems. The figures include a brief written summary of the graphs, describing the graphs and their significance.

These graphs are part of the quarterly report that is distributed to all Information Service management, from Assistant Directors and up, throughout the MTA and the agencies it serves.

4.1 THE PERFORMANCE GRAPHS

The first set of graphs, Figure 2, shows the processor utilization for each system by hour of day. The source data is the HARCPU file. The graphed values are the average over all days of the quarter as well as the maximum and minimum average values observed for each hour during the quarter. They are labeled "AVERAGE", "MAXIMUM" and "MINIMUM", respectively. This profile identifies the daily peaks and valleys of the day, including the usual mid-morning and mid-afternoon peaks and the lunch time valley in between.

Figure 3 shows the processor utilization for each system for each day in the reporting period. The source data is from the HARCPU file. This allows the identification of

heavy days and the worst case hours of each day with the characteristic behavior of these factors.

Figure 4 shows the demand paging for each system for each day in the reporting period. The source data is extracted from the SCPPAG file. This allows for tracking the paging subsystem and, indirectly, the real memory utilization.

Figure 5 shows the aggregate channel utilization for channels with DASD attached. It shows the data, extracted from the HARPCA file, for each system for each day of the quarter.

Figure 6 shows the aggregate DASD volume utilization of the volumes that account for the top 30% of total utilization for each system for each day of the reporting period. The source data is that extracted from the HARDVA file.

4.3 THE CAPACITY GRAPHS

Figure 7 shows two graphs. One is the aggregate processor usage averaged over the Prime Time hours over the quarter for all systems combined. The other is the peak hour average over the quarter for each system. The peak hour usage is what must be supported, while the shift average acts as an indicator of overall usage. It uses data obtained from the HARCPU file.

Figure 8 is the real memory usage during the peak hour of processor usage. To date, this data has come from OMEGAMON/MVS[3] snapshots taken at the peak hour for several days and then averaged.

Figure 9 has two graphs. One shows DASD storage usage compared to installed capacity. the other shows DASD access, the number of actuators available compared to the number needed to provide acceptable performance. The storage data is from the MICS VCC (VTOC/Catalog) component. The access data is estimated using the methodology of J. B. Majors [2].

4.3 THE WORKLOAD GRAPHS

Figure 10 shows the number of batch jobs ending during each day of the reporting period. This is a measure of system throughput.

5. FUTURE DEVELOPMENTS

Presented here is the first cut at this special Performance/Capacity data base and its present limited use of it. In the future, the data in this data base will be used for more extensive reporting. In particular, more detailed workload reporting, such as by agency and application, to track the fine grained temporal usage by agency, as compared to the usual gross monthly usage tracking, often done for accounting purposes. Additional data sources for tracking the on-line and data base facilities and the VTAM network remain to be added to the MICS system and extended to this data base. The capacity parts must be fully automated. Finally, the specific data items stored need to be reviewed and more finely tuned from this initial form.

Bibliography

[1] Hulan E. Jack, Jr., Some Parameters to Evaluate Performance and Capacity om Burroughs B5000/6000/7000 Mainframe Computers, CMG 85 International Conference Proceedings, 1985, pp. 321-327

[2] J.B. Majors, Processor, I/O Path and DASD Configuration Capacity, IBM Systems Journal, Vol. 20, Number One, 1981, pp 63-85.

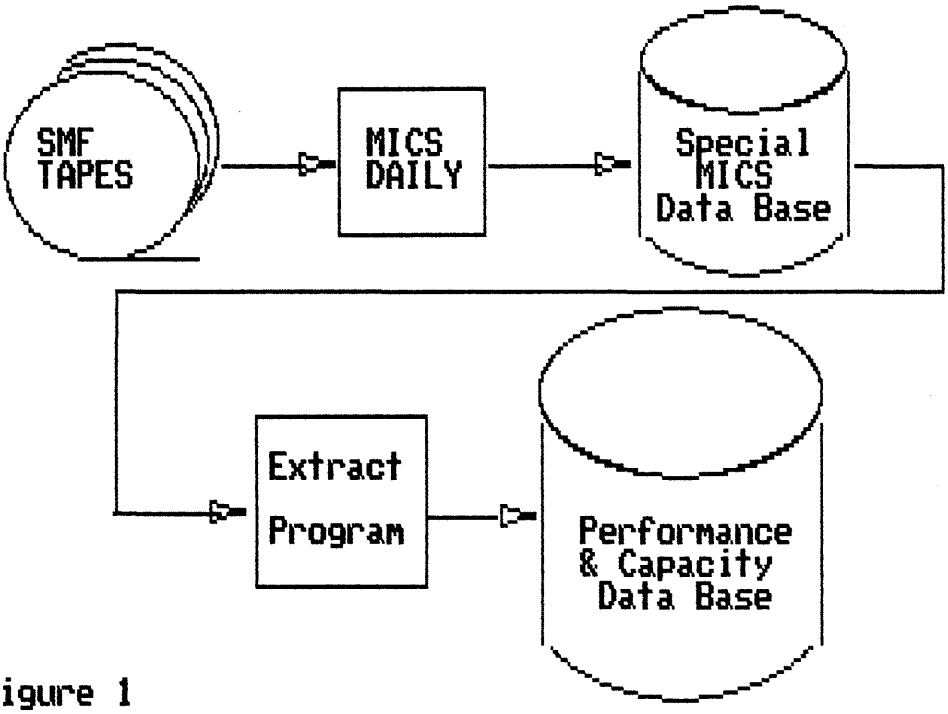


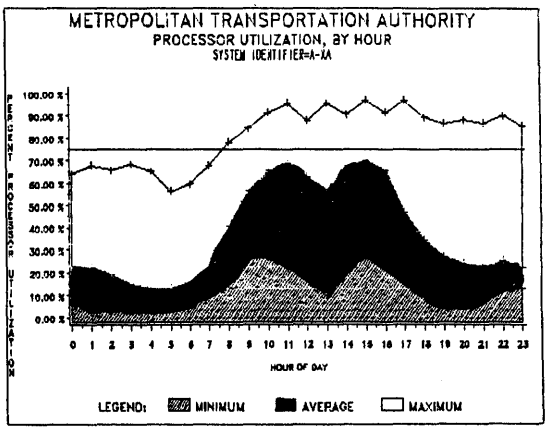
Figure 1

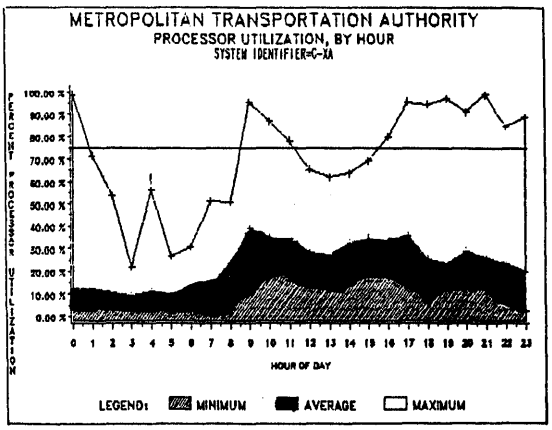
---

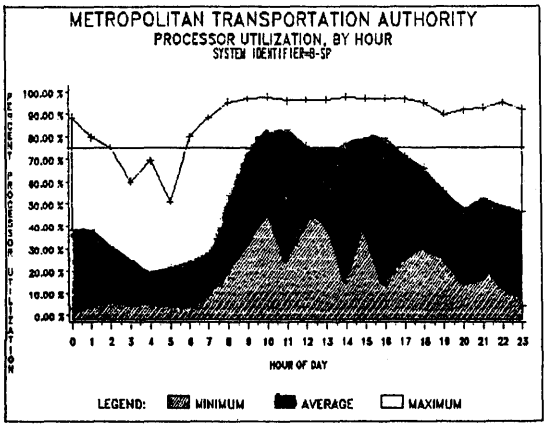
[3]OMEGAMON/MVS is a trademark of Candle Corporation of Los Angeles, CA

Table 1: The DATA Steps for building the Performance/Capacity Data Base

```
***  HARCPU FILE;
DATA  RMFSTA(KEEP= SYSID ENDTS CPUMODEL MICSVER MVSLEVEL
   MVSVER RMFMFL RMFVER STARTTS WEEK ZONE)
      RMFCPU(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   CPU0ONL CPU0CNF CPU1ONL CPU1CNF CPU2ONL CPU2CNF
   CPU3ONL CPU3CNF CPUATT CPUBTT CPUITT CPULTT CPUOTT
   CPURTT CPUSTT CPUTTT CPUWTT
   CPU0BTM CPU0SLIH CPU0TPI CPU0UTM CPU0WTM
   CPU1BTM CPU1SLIH CPU1TPI CPU1UTM CPU1WTM
   CPU2BTM CPU2SLIH CPU2TPI CPU2UTM CPU2WTM
   CPU3BTM CPU3SLIH CPU3TPI CPU3UTM CPU3WTM
   CPUMXA CPUMXB CPUMXI CPUMXL CPUMXO CPUMXR CPUMXS
   CPUMXT CPUMXW CPUNOCPU);

   SET _PHARD.HARCPU01          /* MICS INPUT FILES */
       _PHARD.HARCPU02
       ;

***  HARCUA FILE;
DATA  RMFCUA(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   CUADDR CHANNEL CUAACT CUAATV CUABUS CUACNN CUACUB
   CUACUD CUADIS CUADVB CUAMEC CUAPEN CUAPTH CUAQUE
   CUARSV CUASID CUASSC CUAUTL CUAMSSID CUAMXQUE
   CUANODEV);

   SET _PHARD.HARCUA01          /* MICS INPUT FILES */
       _PHARD.HARCUA02
       ;

***  HARDVA FILE;
DATA  RMFDVA(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   DEVADDR DEVCLASS DEVTYPE CHANNEL CUADDR LCUID VOLSER
   DVAACT DVAATV DVABUS DVACNN DVACUB DVACUD DVADAS
   DVADVB DVAMEC DVAPEN DVAPTH DVAQUE DVARSV DVASID
   DVASSC DVAUTL DVAMSSID DVAMXQUE);

   SET _PHARD.HARDVA01          /* MICS INPUT FILES */
       _PHARD.HARDVA02
       ;

***  HARPCA FILE;
DATA  RMFPCA(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   CHANNEL CHANTYPE CPUID PCAABY PCAOLP PCAORC PCAPRT
   PCASID PCAMPBSY PCAMSSID);

   SET _PHARD.HARPCA01          /* MICS INPUT FILES */
       _PHARD.HARPCA02
       ;

***  SCPLCA FILE;
DATA  RMFLCA(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   LCUID LOGCHAN LCAABY LCACUB LCADVB LCALBC LCALQT
   LCAQRQS LCAQUE LCAPBC LCARQDFR LCARQS LCASIS);

   SET _PSCPD.SCPLCA01          /* MICS INPUT FILES */
       _PSCPD.SCPLCA02


***  SCPLCU FILE;
DATA  RMFLCU(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   LCUATV LCUCNN LCUDIS LCUMEC LCUPEN LCUQRQS LCUQUE
   LCURQDFR LCURQS LCURSV LCUSSC LCUNODEV LCUTOTCH
   LCUTOTCU LCUUTL);

   SET _PSCPD.SCPLCU01          /* MICS INPUT FILES */
       _PSCPD.SCPLCU02
       ;

***  SCPPAG FILE;
DATA  RMFPAG(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   PAGAFC PAGALM PAGAMN PAGASS PAGC PAGDPG PAGDWS PAGEXS
   PAGF PAGFRA PAGLF PAGLFS PAGLP PAGLS PAGMNF PAGMSDPG
   PAGMSSPG PAGMSSWP MAGMXF PAGMXTFC PAGP PAGPMV PAGQ
   PAGR PAGRSS PAGS PAGSIN PAGSOT PAGSPG PSGSSQ PAGT
   PAGTFC PAGTOSWP PAGTWS PAGULS PAGX);

   SET _PSCPD.SCPPAG01          /* MICS INPUT FILES */
       _PSCPD.SCPPAG02
       ;

***  SCPPGA FILE;
DATA  RMFPGA(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   PERFGRP PERFGRPN PGAADJ PGACLS PGACPUTM PGACSD PGAISD
   PGALEVPD PGAMSD PGASSD PGAACT PGACTS PGAITS PGAMTS
   PGASER PGASPP PGASRBTM PGASTS PGATAT PGATCBTM PGATTM
   PGATTX);

   SET _PSCPD.SCPPGA01          /* MICS INPUT FILES */
       _PSCPD.SCPPGA02
       ;

***  SCPPSD FILE;
DATA  RMFPSD(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   PSDDSN DEVADDR DEVCLASS DEVTYPE PSDDS PSDPGX PSDPST
   PSDSIO PSDSLA PSDU PSDUSE PSDMXSTM);

   SET _PSCPD.SCPPSD01          /* MICS INPUT FILES */
       _PSCPD.SCPPSD02
       ;

***  HARTCA FILE;
DATA  RMFTCA(KEEP= SYSID ENDTS DURATION INTERVLS SAMPLES
   TCACHA TCALCA TCA0CB TCA0WB TCA1CB TCA1WB);

   SET _PHARD.HARTCA01          /* MICS INPUT FILES */
       _PHARD.HARTCA02
       ;

***  BATJOB FILE;
DATA  SMFJOB(KEEP= SYSID ENDTS ACCTNO1-ACCTNO3 JOBCLASS
   JOBPRTY PERFGRP RDRTS STATRTS WEEK ZONE
   JOBMASK JOBACCTM JOBBLKTR JOBCPUTM JOBEDASD JOBENQTM
   JOBETAPE JOBEXCPS JOBEXCTM JOBICPTM JOBINQTM JOBNLR
   JOBNSTMT JOBRESTM JOBSTMT JOBTCBTM JOBTCNT JOBTURTM
   JOBXPR JOBMXWSS);

   SET _PBATX.BATJOB01          /* MICS INPUT FILES */
       _PBATX.BATJOB02
       ;
***  BATPGM, BAT_ST AND BAT_TS FILE;
DATA  SMFPGM(KEEP= SYSID ENDTS ACCTNO1-ACCTNO3 PGMTYPE
PERFGRP PROGRAM
   RDRTS STATRTS WEEK ZONE
   PGMSTBOT PGMSTTOP PGMACTTM PGMBLKTR PGMCPUTM PGMEDASD
   PGMENQTM PGMETAPE PGMEXCPS PGMEXCTM PGMICPTM PGMNSTMT
   PGMPGSEC PGMRESTM PGMSTMT PGMTCBTM PGMTCNT PGMMXWSS);

   SET _PBATX.BATPGM01          /* MICS INPUT FILES */
       _PBATX.BATPGM02
       _PBATX.BAT_TS01
       _PBATX.BAT_TS02
       _PBATX.BAT_ST01
       _PBATX.BAT_ST02
       ;
```







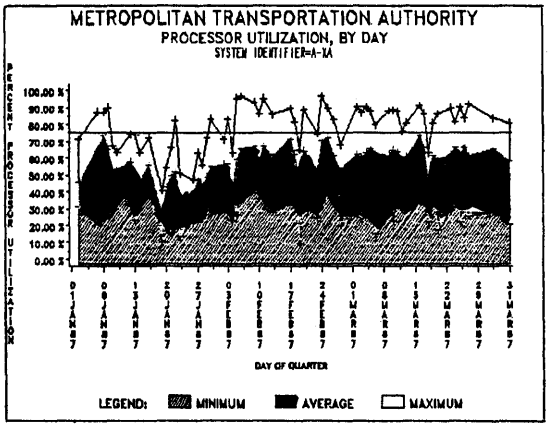
PROCESSOR UTILIZATION, BY HOUR

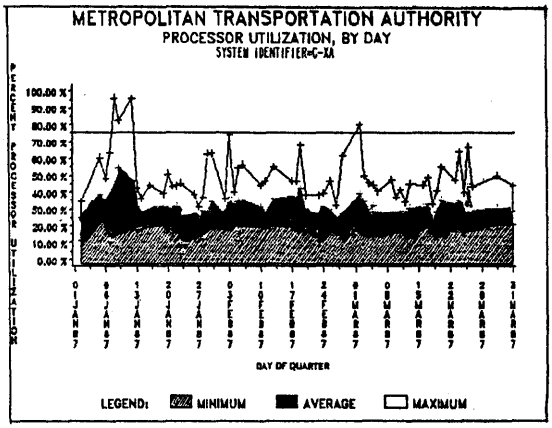
THE GRAPHS SHOW THE MINIMUM, AVERAGE, AND MAXIMUM CPU UTILIZATION BY SYSTEM OVER THE QUARTER. THE AVERAGE FOR EACH HOUR IS THE CPU UTILIZATION FOR THAT HOUR AVERAGED OVER ALL WORKING DAYS OF THE QUARTER. THE MINIMUM AND MAXIMUM VALUES ARE THE LOWEST AND HIGHEST VALUES, RESPECTIVELY, OBSERVED FOR THAT HOUR OVER ALL WORKING DAYS IN THE QUARTER.
RECOMMENDED THRESHOLD: 75% UTILIZATION

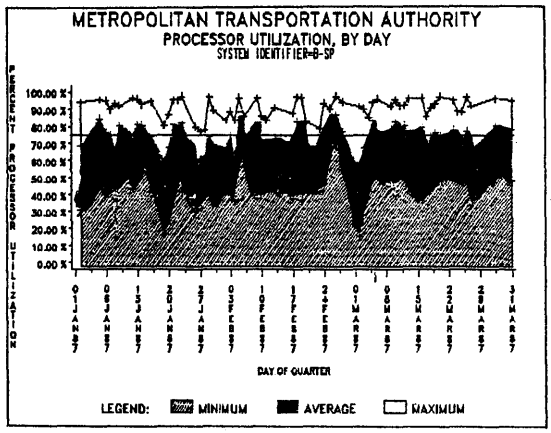
HIGHLIGHTS

THE PEAK HOUR HAS STAYED AT 10-11 A.M., EXCEPT THE C-SYSTEM HAS ITS PEAK BETWEEN 8-9 A.M. IT IS THIS HOUR WHICH IS USED TO PROJECT PEAK CAPACITIES AND LOADING.

FIGURE 2







PROCESSOR UTILIZATION, BY DAY

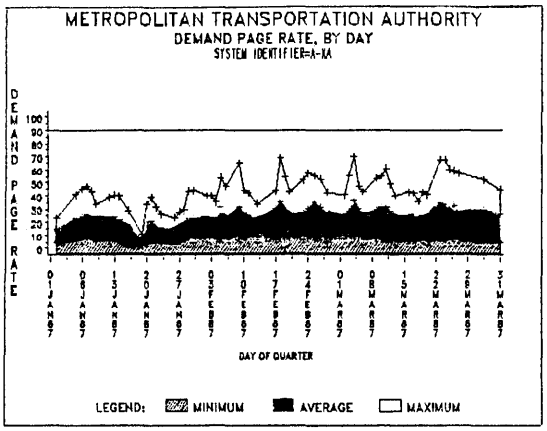
THE GRAPHS SHOW THE MINIMUM, AVERAGE, AND MAXIMUM CPU UTILIZATION BY SYSTEM FOR THE PRIME TIME SHIFT OF THE DAYS OVER THE QUARTER. THE PRIME TIME SHIFT IS 8 A.M.-6 P.M. FROM MONDAY TO FRIDAY. THE AVERAGE IS THE CPU UTILIZATION AVERAGED OVER VALUES OF THE PRIME TIME SHIFT FOR EACH WORKING DAY IN THE QUARTER. THE MINIMUM AND MAXIMUM VALUES ARE THE LOWEST AND HIGHEST HOURLY VALUES, RESPECTIVELY, OBSERVED DURING THAT DAY'S PRIME TIME. FOR PLANNING PURPOSES, THE AVERAGE CPU UTILIZATION IS TRACKED IN ORDER TO ENSURE THAT IT DOES NOT EXCEED THE INSTALLATION THRESHOLD.
RECOMMENDED THRESHOLD: 75% UTILIZATION

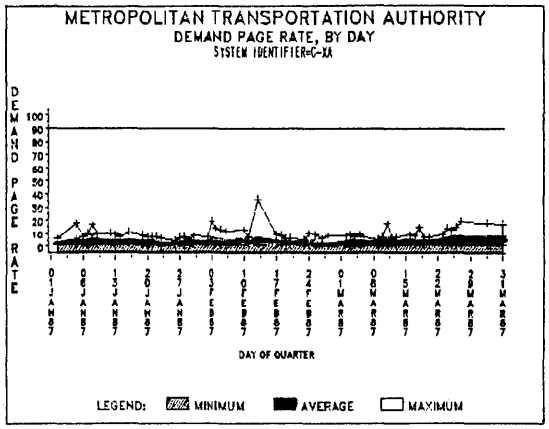
HIGHLIGHTS

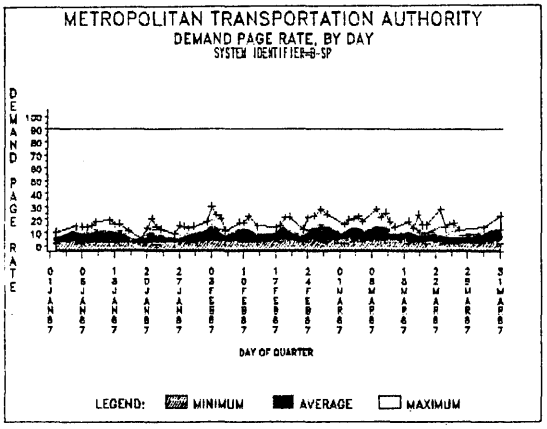
- THE INCREASED UTILIZATION ON THE SYSTEM B HAS CAUSED BY THE ADDITION OF THE NYCTA HUMAN RESOURCES SYSTEM IDMS DATABASE REGIONS, INCREASED TSO AND, IDMS USAGE FOR OTHER NYCTA AND LIRR WORK. THIS HAS CAUSED AN IMBALANCE IN UTILIZATIONS BETWEEN THE A AND B MACHINES: THE SHIFT IS TO THE B . CORRECTIVE LOAD BALANCING WILL BE DONE AFTER THE INSTALLATION OF THE NEW 3090-200 PROCESSOR IN THE EARLY PART OF THE SECOND QUARTER.

- CPU UTILIZATION ON SYSTEM C HAS REMAINED RELATIVELY STABLE IN THE 30% TO 40% RANGE . THIS VALUE WILL INCREASE AS MORE ONLINE WORK IS OFFLOADED FROM THE A AND B SYSTEMS ONTO THE C SYSTEM.

FIGURE 3







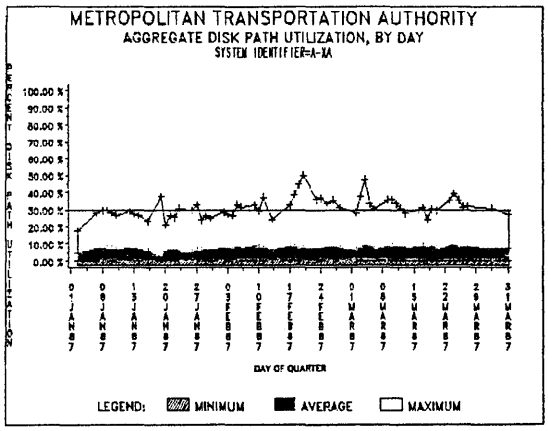
DEMAND PAGING RATE, BY DAY

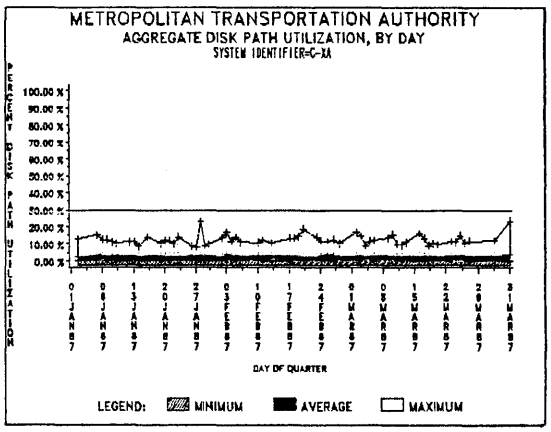
THE GRAPHS DEPICT THE MINIMUM, AVERAGE, AND MAXIMUM DEMAND PAGING RATES (PAGES/SEC) BY SYSTEM FOR THE PRIME TIME SHIFT OF THE DAYS OVER THE QUARTER. THE PRIME TIME SHIFT IS 8 A.M.-6 P.M. FROM MONDAY TO FRIDAY. THE AVERAGE IS THE DEMAND PAGING RATE AVERAGED OVER VALUES OF THE PRIME TIME SHIFT FOR EACH WORKING DAY IN THE QUARTER. THE MINIMUM AND MAXIMUM VALUES ARE THE LOWEST AND HIGHEST HOURLY VALUES, RESPECTIVELY, OBSERVED DURING THAT DAY'S PRIME TIME. THE SYSTEM SUPPORTS USERS WHOSE MEMORY REQUIREMENTS FAR EXCEED THE AMOUNT OF REAL MEMORY CAPACITY OF THE SYSTEM. THIS IS ACCOMPLISHED BY RETAINING IN REAL MEMORY ONLY THAT AMOUNT OF THE USERS' PROGRAM CODE AND DATA THAT WILL SIMULTANEOUSLY ALLOW THEM ALL TO GET INTO THE SYSTEM AND TO RUN EFFICIENTLY. THIS PORTION OF A USER'S REAL MEMORY USAGE IS CALLED THE WORKING SET. WHEN PROGRAM CODE OR DATA IS NEEDED THAT IS NOT IN MEMORY, THE SYSTEM TRANSFERS THEM FROM AUXILLIARY STORAGE (DISK) TO REAL STORAGE IN CHUNKS OF 4K PAGES. THIS IS CALLED DEMAND PAGING. DEMAND PAGING IS GOOD INDICATOR OF CONTENTION FOR MEMORY.
RECOMMENDED THRESHOLD: 90 PAGES/SEC

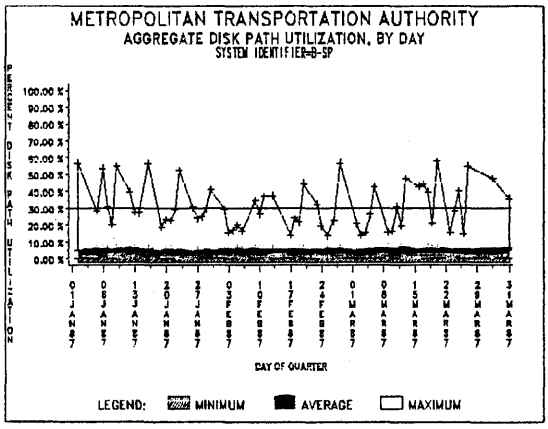
HIGHLIGHTS

- ON ALL THREE SYSTEMS, DEMAND PAGES RATES ARE WELL BELOW THE THRESHOLD OF 90 PAGES/SEC. THESE LOW RATES ARE DUE PRIMARILY TO THE ADDITION OF REAL MEMORY TO THE A AND B SYSTEMS IN THE SECOND QUARTER OF 1986, AND HAVE REMAINED STABLE SINCE THIS UPGRADE.

FIGURE 4







AGGREGATE DISK PATH UTILIZATION, BY DAY

THE GRAPHS PRESENT THE MINIMUM, AVERAGE, AND MAXIMUM DISK PATH UTILIZATION BY SYSTEM FOR THE PRIME TIME SHIFT OF THE DAYS OVER THE QUARTER. THE PRIME TIME SHIFT IS 8 A.M.-6 P.M. FROM MONDAY TO FRIDAY. MINIMUM AND AVERAGE VALUES WERE DERIVED FROM THE AVERAGED UTILIZATIONS OF ALL DISK PATHS ON A GIVEN SYSTEM; THE MAXIMUM VALUE REPRESENTS THE HEAVIEST-UTILIZED DISK PATH FOUND DURING THAT DAY.

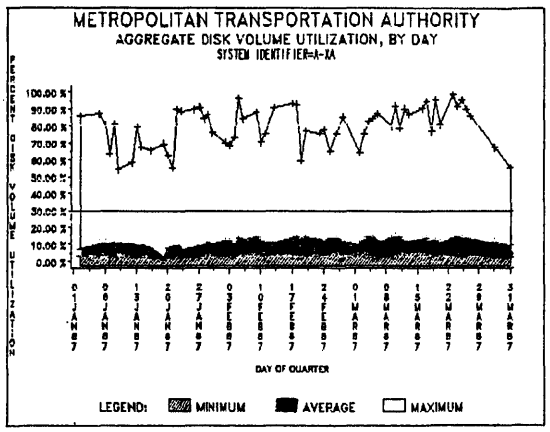
A DISK PATH CONSISTS OF A CHANNEL PATH/DISK CONTROL UNIT COMBINATION WHICH CONNECTS EACH BANK OF DISK DEVICES TO THE CPU. THE PATH IS BUSY WHENEVER DATA IS TRANSFERRED TO OR FROM DISK. PATH UTILIZATION IS THUS A MEASURE OF THE I/O INTENSITY OF THE WORKLOADS RUNNING ON A GIVEN SYSTEM.
RECOMMENDED THRESHOLD: 30% UTILIZATION

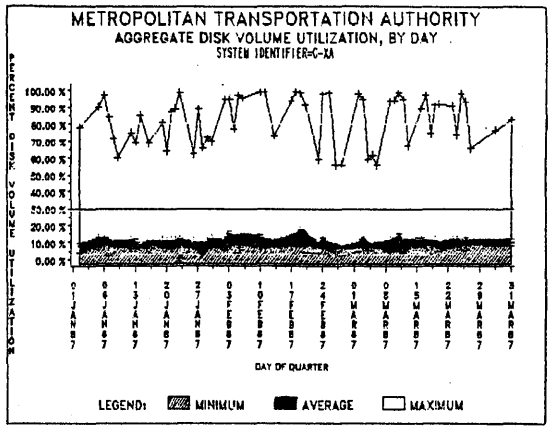
HIGHLIGHTS

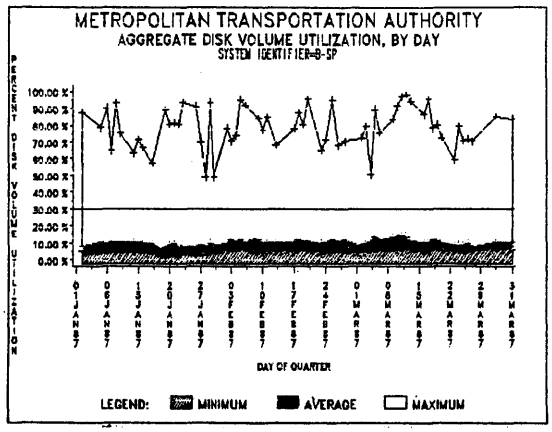
- AVERAGE PATH UTILIZATION IS WELL BELOW THE 30% UTILIZATION THRESHOLD.
- THE MAXIMUM PATH UTILIZATION IS CAUSED MOSTLY BY PAGING AND SWAPPING ACTIVITY. ALTHOUGH UTILIZATIONS ABOVE 30% ARE NOT RECOMMENDED FOR PATHS SERVICING USER REQUESTS, PATHS SERVING SWAP REQUESTS USUALLY EXHIBIT HIGH UTILIZATIONS WITHOUT ADVERSELY IMPACTING OVERALL SYSTEM PERFORMANCE. THUS, HIGH SWAP PATH UTILIZATION IS NOT A PROBLEM IN AND OF ITSELF. HOWEVER, TWO FACTORS SHOULD BE NOTED:
  - SOME OF THE PLOTTED MAXIMA CONSIST OF PATHS SERVICING USER REQUESTS;
  - SWAP DATASETS ARE SHARING PATHS WITH USER I/O.

CONCLUSION: DASD I/O IS GRADUALLY BECOMING A BOTTLENECK TO SYSTEM PERFORMANCE AND REQUIRES TUNING.

FIGURE 5







AGGREGATE DISK VOLUME UTILIZATION, BY DAY

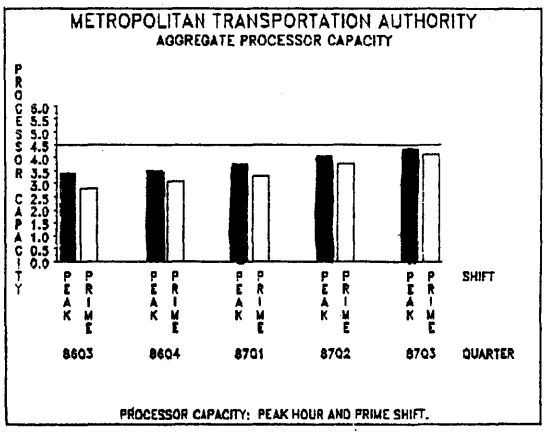
THE GRAPHS DISPLAY THE MINIMUM, AVERAGE, AND MAXIMUM DISK VOLUME UTILIZATION BY SYSTEM FOR THE PRIME TIME SHIFT OF THE DAYS OVER THE QUARTER. THE PRIME TIME SHIFT IS 8 A.M.-6 P.M. FROM MONDAY TO FRIDAY. THE DATA FOR THESE ANALYSES ARE DASD VOLUMES WHICH ACCCOUNT FOR 80% OF THE ACTIVITY. THE MINIMUM AND MAXIMUM VALUES REPRESENT THE LOWEST AND HIGHEST HOUR DETECTED FOR THAT DAY. MINIMUM AND AVERAGE VALUES WERE DERIVED FROM THE AVERAGED UTILIZATIONS OF THE TOP 30% OF VOLUMES ON A GIVEN SYSTEM; THE MAXIMUM VALUE REPRESENTS THE HEAVIEST-UTILIZED DISK VOLUME FOUND DURING THAT DAY.
RECOMMENDED THRESHOLD: 30% UTILIZATION

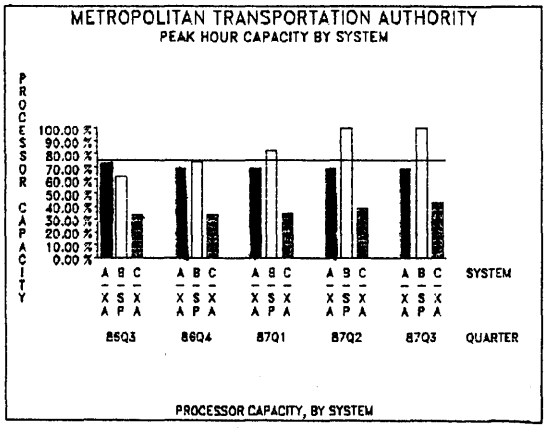
HIGHLIGHTS

- AVERAGE VOLUME UTILIZATION IS WELL BELOW THE 30% UTILIZATION THRESHOLD
- A FEW DISK VOLUMES ARE EXHIBITING HEAVY I/O ACTIVITY. THE BULK OF THESE VOLUMES ARE SYSTEM PACKS CONTAINING SYSTEM LIBRARIES, CATALOGS, AND PAGING/SWAPPING FILES. IN ADDITION, SEVERAL TSO VOLUMES ARE BEING HEAVILY UTILIZED. SYSTEM VOLUMES WILL BE EXCLUDED IN LATER REPORTS IN ORDER TO CONCENTRATE MORE READILY ON I/O GENERATED BY USER TASKS. ALTHOUGH THE PLOTTED MAXIMA APPEAR SOMEWHAT FRIGHTENING, IT SHOULD BE NOTED THAT THESE POINTS REPRESENT A FEW, HEAVILY-UTILIZED VOLUMES - MOSTLY SYSTEM AND WORK PACKS, AND SOME USER VOLUMES (E.G., TSO).

CONCLUSION: DASD I/O IS GRADUALLY BECOMING A BOTTLENECK TO SYSTEM PERFORMANCE. REDISTRIBUTION OF DATASETS OVER THE PACKS IS NEEDED. SUCH ACTION REQUIRES THE USE OF ADEQUATE MEASURING TOOLS.

FIGURE 6


METROPOLITAN TRANSPORTATION AUTHORITY
AGGREGATE PROCESSOR CAPACITY
PROCESSOR CAPACITY
PEAK PRIME SHIFT
86Q3 86Q4 87Q1 87Q2 87Q3 QUARTER
PROCESSOR CAPACITY: PEAK HOUR AND PRIME SHIFT.



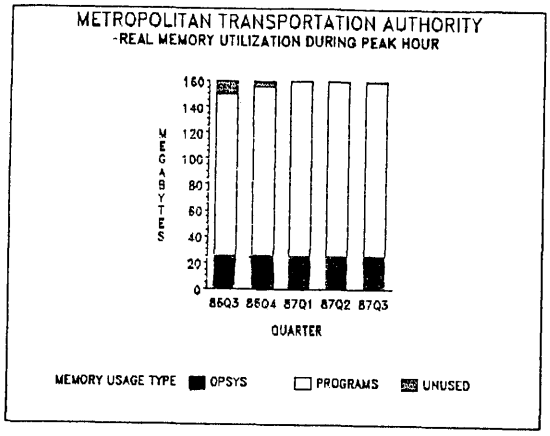
AGGREGATE CPU CAPACITY

THIS GRAPH DEPICTS THE INSTALLED, USED, AND AVAILABLE CPU CAPACITY OF THE MTA DATA CENTER OVER FIVE QUARTERS - THE PREVIOUS TWO, THE PRESENT, NEXT TWO. THE NEXT 2 QUARTERS WERE DERIVED BY EXTRAPOLATING THE GROWTH SHOWN OVER THE FIRST THREE QUARTERS. THE EFFECTIVE CAPACITY IS 75% OF INSTALLED CAPACITY, OR 4.5 CPU HOURS OUT OF THE TOTAL CPU CAPACITY OF 6 CPU PER ELAPSED (WALLCLOCK) HOUR.
CPU UTILIZATION IS GROWING SLIGHTLY SLOWER THAN PREVIOUSLY PROJECTED. THIS IS THE THIRD QUARTER 1986 LOAD DUE TO NYCTA BUDGET DATA ENTRY (BDES) BATCH WORK. THIS LOAD WILL, HOWEVER, RETURN IN THE THIRD AND FOURTH QUARTERS OF 1987. THE UPGRADE DUE IN THE SECOND QUARTER OF 1987 OF REPLACING THE IBM 3081KX PROCESSOR WITH AN IBM 3090-200 PROCESSOR INCREASES THE AGGREGATE PROCESSOR POWER BY ABOUT 33%. THIS SHOULD PRECLUDE CONCERNS FOR CPU HORSEPOWER FOR AT LEAST THE NEXT TWO OR THREE QUARTERS.

PEAK HOUR CPU CAPACITY BY SYSTEM

THIS GRAPH SHOWS THE CPU UTILIZATION OF EACH SYSTEM OVER FIVE QUARTERS - THE PREVIOUS TWO, THE PRESENT, AND THE NEXT TWO. THE VALUES FOR THE NEXT TWO QUARTERS ARE DERIVED BY EXTRAPOLATING THE GROWTH SHOWN IN THE FIRST THREE QUARTERS.
DURING THE SECOND AND THIRD QUARTERS OF 1986, A LARGE AMOUNT OF WORK WAS MOVED FROM SYSTEM "A" TO SYSTEM "B". THIS ACTION RESULTED IN A MORE BALANCED UTILIZATION BETWEEN THESE TWO MACHINES. INCREASED DEVELOPMENT ACTIVITIES ON THE "B" SYSTEM HAVE DESTROYED THIS BALANCE. THE OVERUTILIZATION OF THE "B" SYSTEM WILL CONTINUE THROUGH THE SECOND QUARTER OF 1987. AS THE NEW IBM 3090-200 PROCESSOR IS INSTALLED AND "BURNS IN" DURING THE SECOND QUARTER, THE WORKLOAD WILL BE APPROPRIATELY REDISTRIBUTED OVER THE THREE SYSTEMS TO ACHIEVE BALANCE.
BECAUSE OF THE EXPECTED UPGRADE OF THE IBM 3090-200 REPLACING THE IBM 3081KX, AND THE A-SYSTEM WORKLOAD MOVING FROM THE ONE-HALF IBM 3084QX TO THE NEW PROCESSOR, THE PROJECTIONS FOR THE A-SYSTEM UTILIZATION SHOULD BE REDUCED TO HALF OF THEIR PRESENTLY SHOWN HEIGHTS. THIS IS BECAUSE THE TWO CPU ENGINES ON THE IBM 3090-200 HAVE ABOUT TWICE THE COMPUTING POWER AS THAT OF THE TWO CPU ENGINES ON ONE-HALF OF AN IBM 3084QX.

FIGURE 7



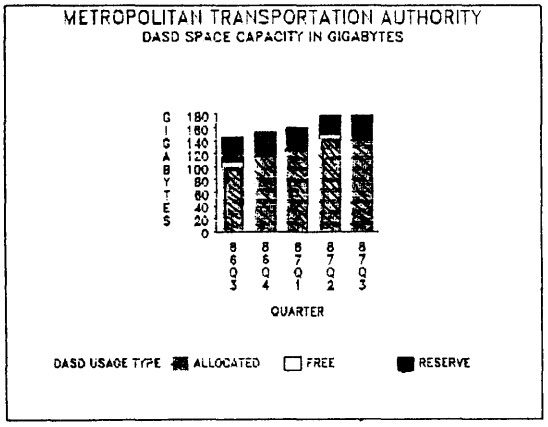
REAL MEMORY UTILIZATION DURING PEAK HOUR

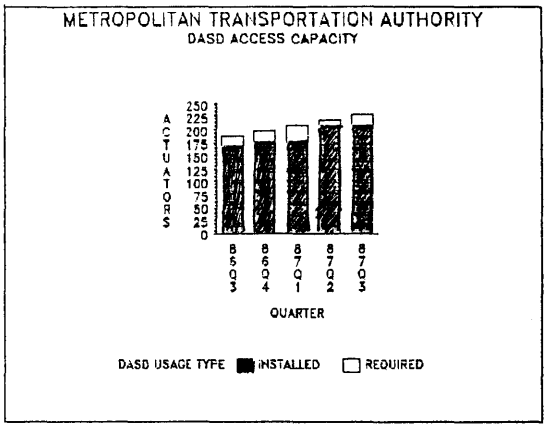
THIS GRAPH DEPICTS A MODEL SNAPSHOT OF REAL MEMORY UTILIZATION DURING THE PEAK HOUR (10-11 A.M.). THE BULK OF MEMORY IS USED BY APPLICATIONS, ESPECIALLY ONLINE IDMS AND CICS REGIONS; REMAINING MEMORY IS SPLIT BETWEEN OPERATING SYSTEM (MVS NUCLEUS, ACCESS METHODS, ETC.) AND UNUSED FRAMES.

MEMORY UTILIZATION IS GREATER THAN ANTICIPATED BECAUSE PREVIOUS SCENARIOS ASSUMED THE USE OF THE 3084 IN SINGLE-IMAGE MODE (ONE COPY OF THE OPERATING SYSTEM RUNNING BOTH SIDES OF THE 3084) INSTEAD OF THE CURRENT PARTITIONED MODE (EACH SIDE UNDER CONTROL OF SEPARATE COPIES OF THE OPERATING SYSTEM). THE IMPLEMENTATION OF SINGLE-IMAGE MODE, DELAYED DUE TO SOFTWARE PROBLEMS, WILL ELIMINATE ONE COPY OF THE OPERATING SYSTEM AND FREE APPROXIMATELY SIX TO SEVEN MEGABYTES.

THE FULL MEMORY IS NOW BEING TOTALLY USED. FROM THIS POINT ON PAGING CAN BE EXPECTED TO INCREASE AS ONLINE SYSTEMS WITH THEIR INCREASED MEMORY DEMANDS CONTINUE TO GROW. THIS EXPECTATION OF INCREASED PAGING MAY NOT BE FULLFILLED DUE TO THE DIFFERENT ARCHITECTURE OF THE IBM 3090-200 DUE

FIGURE 8





DASD SPACE CAPACITY IN GIGABYTES

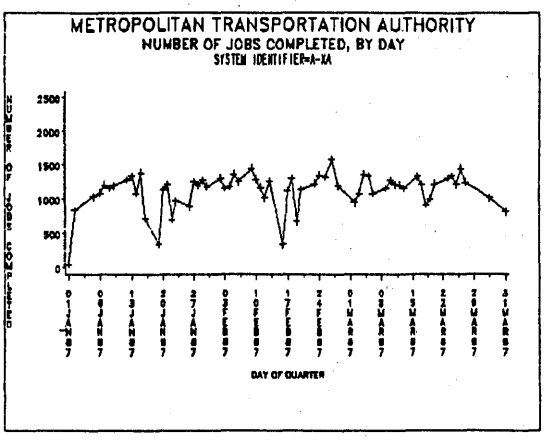
CURRENT PLANNING CALLS FOR A MAXIMUM EFFECTIVE DASD SPACE UTILIZATION OF 75% OF INSTALLED CAPACITY (I.E., 25% OF INSTALLED CAPACITY FOR RESERVE, FILE GROWTH, ETC.). THE FREE SPACE REPORTED HERE IS THE TOTAL INSTALLED MINUS THE AMOUNT ALLOCATED MINUS THE 25% OF INSTALLED CAPACITY FOR RESERVE. IF THIS NUMBER FALLS BELOW ZERO, IT IS REPORTED AS ZERO AND THE RESERVE IS REDEFINED AS THE TOTAL INSTALLED MINUS THE AMOUNT ALLOCATED. THE CURRENT INSTALLED DASD CAPACITY IS ABOUT 161 GIGABYTES (A GIGABYTE IS ONE BILLION BYTES); EFFECTIVE CAPACITY IS THEREFORE 120 GIGABYTES.

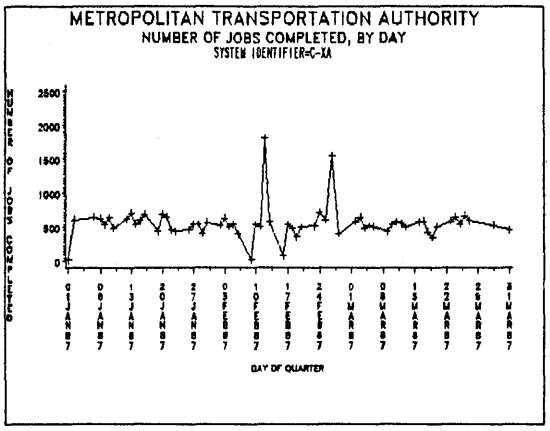
THE EFFECTIVE CAPACITY IS EXHAUSTED. BUT RELIEF IS IMMINENT BECAUSE 8 IBM 3380 DASD DEVICES (32 ACTUATORS AND 40 GIGABYTES) ARE DUE TO BE DELIVERED ABOUT JUNE 1, 1987 UPON COMPLETION OF THE NEW DASD ROOM. THIS SHOULD MEET OUR STORAGE SPACE NEEDS FOR THE THIRD AND FOURTH QUARTERS OF 1987. THE SECOND QUARTER 1987 STORAGE WILL, HOWEVER, BE TIGHT.

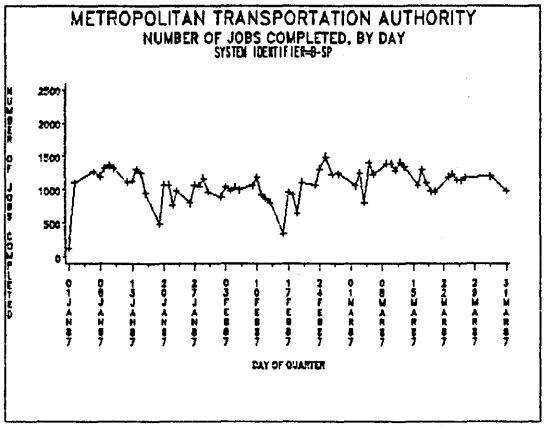
DASD ACCESS CAPACITY

"ACCESS CAPACITY" IS A TERM WHICH REFERS TO THE NUMBER OF ACTUATORS - DISK ARMS - REQUIRED TO SERVICE THE I/O RATES GENERATED BY THE GROWING WORKLOAD. AS SHOWN, THE MTA HAS APPARENTLY EXCEEDED INSTALLED ACCESS CAPACITY SINCE THE THIRD QUARTER OF 1986. I/O SUBSYSTEM TUNING WOULD BUY BACK ABOUT 15% TO 20% OF THE ACCESS CAPACITY. SOME DEVICES AND PATHS ARE PRESENTLY UNDERUTILIZED WHILE OTHER ARE OVERUTILIZED. OTHERWISE ABOUT 32 ADDITIONAL ACTUATORS ARE NEEDED TO GUARANTEE GOOD PERFORMANCE. 32 MORE ACTUATORS ARE DUE TO BE INSTALLED ABOUT JUNE 1, 1987 AFTER COMPLETION OF THE NEW DASD ROOM. THAT CAPACITY WILL BE EXCEEDED AS QUICKLY AS IT IS INSTALLED. TUNING BY REDISTRIBUTING DATA SETS OVER THE VOLUMES WOULD BUY BACK AT LEAST ABOUT 20% OF THE ACCESS CAPACITY. DASD ACCESS CAPACITY, ALONG WITH THE GIGABYTE PROJECTIONS, DEFINE THE AMOUNTS OF DASD WHICH ARE TO BE REQUIRED TO SUPPORT THE WORKLOAD.

FIGURE 9







WORKLOAD GRAPHS

NUMBER OF JOBS COMPLETED, BY DAY

THE GRAPH SHOWS THE NUMBER OF BATCH JOBS COMPLETED DURING EACH 24 HOUR DAY OF THE QUARTER FOR EACH OF THE THREE SYSTEMS.
RECOMMENDED THRESHOLD: N/A

FIGURE 10

FINANCIAL CONSIDERATIONS IN CAPACITY PLANNING

Jerry L. Rosenberg
SRM Associates, Ltd.

ABSTRACT

The corporate decision to acquire additional data processing equipment is usually based on multiple considerations. The most critical of these is often the financial aspects of the available configuration options. While capacity planners have become more sophisticated in the methods of selecting the best technical configuration, we have been slow to address the budgetary aspects of our recommendations.

This paper will point out the topics that should be considered and presented to develop more complete hardware recommendations. The author will discuss the key areas of financial concern and how they affect purchase decisions. The hidden costs of new equipment will be covered. These will include site preparation, software licenses, personnel costs, conversion efforts, peripheral equipment, and cost of money.

A case study will illustrate the use of a standard PC based spreadsheet package for analyzing costs and determining present costs of equipment. The author will use this technique to develop a total cost of a configuration by year and show how to represent the real yearly dollar cost of each alternative.

Before we enter into a discussion of the content and nature of the financial analysis that capacity planners should perform, we should first consider what might be a common question - Why should capacity planners worry about monetary issues at all?. To best address this question, a definition of the goals of capacity planning must be established.

The goal of capacity planning, as I see it, is to insure that the proper configuration is in place to address the data processing requirements of the enterprise in sufficient time to maintain satisfactory levels of service to the user community in a cost-effective manner.

This goal is not accomplished when the right technical solution is determined. It requires that the proper configuration be justified, acquired and implemented in a time-sensitive manner. Figure 1 represents this view of capacity planning. It must be an interface function that incorporates and addresses the technical aspects of the data processing environment and staff with the service needs and expectations of the user communities. In addition, it must be understood that these two often diverse groups are further influenced by the overall demands and goals of the corporation. These high level goals are usually related to and expressed in the desire to increase profitability. It is vital for the capacity planner to accept and understand that most bottom-line decisions in the corporation will be made based on profitability and return on investment. The logical approach is to not fight the tide but rather to go with the flow and utilize the financial aspects of configuration analysis to our own advantage.

As an added incentive, I would ask capacity planners to consider whether they have ever submitted a plan to management that was well developed, artfully presented and technically correct but which was not followed for budgetary or other financial reasons. The most common complaint I hear from capacity planners is - They ask me to do the study, then they don't listen to me anyway. This counter-productive frustration is avoidable. You really cannot expect anyone to respond to you if you do not speak to them in the language that they understand. Let's face it, businessmen talk money, and ultimately it's businessmen who make the final decision concerning data processing acquisitions.

If you have agreed with the discussion so far (or, at least, not discounted it totally), let's continue with a very high-level overview of the contents of a capacity study. My suggestions for contents of a capacity presentation would include:

- o Statement of the objective of the study
- o Methodology used
- o Hardware and software alternatives
- o Benefits/exposures of each alternative
- o Lifespan of each alternative
- o Impact of not adopting each approach
- o Cost (current and future) of each alternative
- o A recommended approach

The remainder of this paper will deal with the presentation of cost factors. As a first step, let's do some mental stretching exercises to prepare us for this activity. As can be seen, many of the other sections of the presentation are technical in nature, while the cost issue has to be more business or management in scope. To begin to make the transition, we can view the concepts and terminology that are inherent to each group:

Technical Concepts	Management Concepts
Monitors	Planning
CPU	Budgeting
I/O	Chargeback
Memory	User Service
Benchmarks	Cost of Service
Modelling	Return on Investment
Statistical Analysis	Useful Life

It should be no surprise that corporate management expects to see an analysis and recommendation that addresses and that is couched in the issues that are listed in column two. Their goals are optimizing profits and minimizing costs. If we, as capacity planners, want to get their attention, we have to deal in these issues.

Many of us in the technical arena have a difficult time even beginning this process of financial analysis. For those who require a checklist to help them through the transition period, I offer the following steps that can be taken to help one become more sensitive to the financial aspects of data processing.

- o Understand the primary business of the firm
- o Understand how computing supports the primary business
- o Start thinking and talking about profitability
- o Discuss both costs and return on investment
- o Don't be afraid of financial terms and concepts

As you get more comfortable in dealing with the financial issues, you can incorporate the following guidelines:

- o Wherever possible, equate costs to potential returns. What this really means is, always try to present the positive side of the picture. It is best to show data processing acquisitions as investments rather than costs.

- o Relate your analysis to the corporate bottom line and the budget. Always attempt to illustrate how the recommended approach will aid other departments in the fulfillment of their goals. Point out how this effort will lead to increased corporate profitability.

- o Discuss the costs of not meeting service level objectives. These will include:
  - Lost sales due to inadequate service
  - Lost revenue due to the inability to react or due to lack of knowledge
  - Employee Dissatisfaction
  - Inhibited Business Growth
  - Violation of Service Level Contracts
  - Legal Problems as a result of Non-performance

Figure 2 summarizes the point I am trying to make. Sooner or later all of us have to accept that the monetary issues must be addressed to insure a satisfactory outcome to the capacity planning process. Hopefully, the reader agrees with this proposal and we can continue on to the discussion of those factors which make up configuration cost that should be analyzed and presented.

Cost Factors in Configuration Analysis

How does one go about including financial analysis in a capacity study ? First, we should begin by understanding all of the cost factors that apply to configurations. Just as we accept that sizing a CPU does not constitute a complete evaluation, we must understand that there are more financial considerations than just the lease vs. buy concern. In the technical analysis we are astute enough to include the necessary ancillary equipment required to create a total configuration. In the financial analysis we must discuss all of the potential monetary exposures. At a macro level they include:

- o Hardware
- o Software
- o People
- o Facility
- o Supplies
- o Overhead
  - Indirect Costs
  - General Business Expenses

The paper will discuss each of these topics and itemize some of the concerns that should be addressed. In addition, as a second level topic set, the following concepts will also be covered:

- o Direct Costs
- o Cost of Early Disposition
- o Conversion Costs
- o Life Expectancy
- o Supportive Equipment
- o Licenses

In most cases, these issues can result in savings rather than costs. If this is the case, be very sure to bring this to light. Let's discuss each of the major areas in some detail.

Hardware

This issue is the most readily understood component of financial analysis, but it does have some pitfalls. To put the exercise in the right perspective, we should postulate rule #1. The capacity planner is not the corporate financial analyst. He is going to include cost and benefit figures in his proposal but he is not the one who is going to perform lease/buy analysis, estimate the value of money or find the most advantageous third party arrangement.

The capacity planner is only responsible for enumerating all of the costs associated with a given technical solution and making reasonable comparisons between alternate solutions. The key here is to make sure that you are comparing apples to apples. Don't try to estimate the best lease arrangement, just use the same one for all comparisons. What this means is, if you use a purchase price for one configuration, don't compare it to a third party three year lease for another. We are simply trying to establish a relative cost/benefit ratio. It is important that all factors are adjusted to the same base. In essence, we are not attempting to develop an absolute cost of each configuration. That will depend on negotiations and leasing terms. We are trying to establish the relative financial impacts of various configuration alternatives.

When dealing with hardware costs, it is also important to relate to several key components in the cost equation: Supportive Equipment, Early Disposition, and Life Expectancy.

Supportive Equipment

This is an often neglected hidden cost in configuration selection. The CPU alone does not constitute the configuration. Most likely changes to all the peripheral pools will will be required. Each of these must be delineated to produce a complete picture of the requirements of the recommendation. The costs of all components of the suggested configuration must be addressed.

Early Disposition

The capacity planner must be aware that most configurations are financed over a fixed time period. Even if there is no penalty for early lease buy-out, there is still the need to account for the book value of the equipment in the asset columns of the company books. We must either consider this to be an extra cost that is included in the equation or we must find an equitable solution to the disposition problem.

Fortunately, in current times, with the advent of multiple data center corporations, the question is often easily addressed by relocating a piece of equipment from one site to another. The ability to suggest this course of action requires that the capacity planner keep aware of two key pieces of information: the current lease/amortization status of in-house equipment and the capacity requirements of the entire corporation. The effort will be well rewarded. It is always expeditious to have a means of defraying any costs that you might be incurring.

Life Expectancy

This topic is a double edged sword. The tendency might be to over estimate the potential cost of a configuration in order to lower the projected annual cost. Resist the temptation. If you succumb, at the end of the real useful life of the equipment, you will be faced with the early disposition question mentioned above. Except that the equipment that you are disposing of will have a highly inflated book value that you cannot recoup. Spend the time and effort and develop a reasonable life expectancy for the configuration that you propose.

There have been several papers over the past few years that have addressed hardware related financial issues. I

refer the reader to those for future discussions of lease/buy analysis, third party options, etc. These are important issues to consider, but are more complex than the novice financial analyst should attempt. For starters, we are trying to just identify and itemize as many of the costs associate with a given proposed configuration as possible. In addition, we will be concerned with quantifying the benefits (financial) of each component of our recommendation.

Software

The need to convert software or alter existing licensing agreements is often overlooked in developing configuration costs. The licensing issue alone can radically affect monthly charges.

There are some obvious reasons for a software change resulting from a hardware acquisition. Recently, many data centers coordinated conversions to XA to coincide with (or predate) the implementation of new hardware. I suspect that many of them did not represent the software conversion as a necessary cost of the new equipment. It really was. In addition, to make full use of the new environment, new releases of sub-system software (CICS, IMS, etc.) will be installed over time. These should also be addressed. The planner should not avoid these items since it is often easy to offset the costs of conversion with long range benefits. Including all of the cost factors gives management the impression that a thorough job has been done and makes them more confident with the overall recommendation.

The question of software licenses can often work to your advantage. The critical question to ask yourself in reviewing licenses is are they CPU or site based. For those products that are site based (one license permits you to use the product on any CPU at a site) there is probably little change in the costs associated with the product. However, for those components that are priced per CPU (the license allows the use of the product on only one CPU), there is a potential for either cost or savings. If you are making a one for one replacement, It is probable that the costs of software will stay constant (except where new releases have higher costs). If you are replacing CPU's with an increased number of CPU's, then the costs will increase.

The good news is that very often the acquisition of a large CPU is the result of incorporating workloads from several smaller complexes. If this is the case, the company stands a chance of recouping sizable software rental costs for CPU based products. When making comparisons, be sure to investigate how each proposed configuration and workload mix will affect the software license question and include it in your analysis.

People

Whatever the activity, there will be some costs incurred for the expenditure of time and effort. Don't forget them. They can add up. Even if you do not feel comfortable equating the time required to actual dollars for hardware and software efforts, at least mention that these issues exist so that management can consider them.

The proposal should include the efforts of hardware and software implementation, conversion needs, re-education and the possibility of an increase or decrease in real head count. (It is hard to operate two machines with the same staff as you operated one machine.) It is advisable to bring these issues out in the open under your control rather than to allow another department (i.e.. operations management) to use them in countering your recommendation later on.

Facility Requirements

This is another often overlooked area of cost. There is a significant amount of site costs that occur in direct support of the computing environment. These costs vary greatly from configuration to configuration. At a minimum, the following expenditures must be investigated:

- o Floor Space
- o Power
- o Cooling (AC, Chilled Water, etc.)
- o Site Preparation

The increase or decrease in the the cost of each of these can greatly effect the overall cost of the target configuration. In addition, if there is any discussion of possible geographic changes regarding the data center, there can be enormous differences in these costs from one locale to another. The costs of these support components of the data center and the time involved to implement them must be evaluated.

Supplies

On occasion, there will be the requirement to convert some inventory items as a result of a change in the configuration. For example, the decision to acquire cartridge tape units to replace standard reel models will necessitate the purchase of an inventory of new tapes. In addition, it may require a considerable conversion effort in modifying JCL or transferring tape data sets. These costs must be itemized when proposing the new tape units. In all likelihood, the increase in speed and reduction in errors will more than compensate for the expenditure. But we should make that case before someone else puts us on the defensive.

Overhead

All other items that I may have missed can be incorporated into the overhead category. (This is a very useful technique I acquired while performing analytic modeling efforts. Just as in the modeling case, where overhead (or other) represents real CPU or I/O activity, the overhead cost category equates to real dollars. Each corporation will have varying overhead costs that the analyst has not addressed in one of the other areas.

If the indirect costs of personnel (vacation, education, space, insurance) have not been included in the people category, include it now. In addition, general business expenses must be addressed, such as increased insurance costs for plant and equipment, additional phone lines and carrying charges to support added TP activity, new costs due to the delivery of output as a result of a data center move, etc. There can be many costs which will be unique to the particular situation. Simply ask yourself honestly whether you have covered all of the costs and financial impacts of the recommendation that you have made.

As a good frame of reference, remember the analysis that you conducted when you bought your house or your last car. The real issue was the bottom line cost per month. Think in those terms and you will be okay. But don't forget the add on alarm system, undercoating, paint finishing, extended warranty, added cost of insurance, etc. All of those can greatly affect the cost of the automobile you choose (assuming that you have already negotiated the trail of options that brought you from the sticker price to the real price).

Don't make management feel that they are uncovering the real monthly cost one expensive layer at a time. Try to honestly identify all the relevant issues, quantify the dollars associated and justify the expenditures. You will stand a much better chance of getting your recommendations accepted. You will also stand a much

better chance of impressing management with the thoroughness and integrity of your work.

Why?

Why do I, a data processing person, want to go through this exercise ? Data processing does not exist as a unique scientific endeavor but rather as a critical support structure in the corporate enterprise. It is therefore incumbent that data processors relate to the goals of the corporation. Those goals are most often translated and expressed in monetary terms. If we want to live in the corporate realm and personally progress within it, we must learn the language.

Perhaps we should start to retitle our job descriptions to be more in keeping with corporate goals. What we really want to do for the company is resource management not simply capacity management. Capacity management implies that we can measure how much service a particular grouping of hardware and software can deliver and how much bigger that grouping has to be in the future. Resource management also includes an understanding and analysis of the ancillary requirements of the delivery of service, the costs of supplying that service and the net value to the corporation.

Remember, always relate to the net bottom line. Not only what it costs, but what it's worth. If you can illustrate that the the configuration you are proposing is a profitable choice for the company, not only will that proposal be more successful; **so will you.**

Analysis Example

The four year analysis in Figure 3 illustrates a methodology for presenting a financial evaluation of a configuration. This breakdown, with accompanying text, permits an understanding of both yearly and total costs. Supplying this analysis for each configuration that is proposed will allow a quick comparison of the financial implications of each decision.

The dollar values in figure 3 are for illustrative purposes only and are not the real prices of of any particular configuration. In fact, the technique that has been employed makes use of delta values rather than absolute values. The absolute value method would use, as prices, the full market price of each component. What the author has done in the example, is to reflect the change (delta) in cost for each item as compared with the current configuration. Either approach is acceptable and the choice will depend on the financial data available to the analyst and the desired presentation format.

The spreadsheet was generated using LOTUS, a PC based spreadsheet package. The analyst had to supply the assumptions for the useful life of the configuration and the inflation rate (cost of money) per year. In addition, the price delta and one-time costs were supplied. All remaining values were automatically derived through spreadsheet formulas and macros. For those interested in the underlying spreadsheet formulas, please contact me and I will send you a copy.

As mentioned above, the first pieces of data to be supplied are the estimated useful life of the configuration and the anticipated yearly inflation rate that applies to costs. The strength of the spreadsheet package is that the analyst can easily change either or both of these variables and recompute all the values for yearly and total costs. Likewise, any of the pricing estimates can be easily adjusted as required. This gives the analyst the capability of answering some financial what-if's with sensitivity analysis.

The next step would be the establishment of prices for each catagory. In the example, the delta approach was used so the numbers inserted represent the net change (plus or minus) from the current sistuation. I have not considered the issue of residual cost in this example. Conceptually, we either deduct the estimated salvage value of each configuration before computing the delta price. The analyst is invited to consider adding a column to the spreadsheet to enumerate this value if desired. In the example, we are anticipating changes to the CPU, DASD and tape sub-systems, and other equipment (i.e. printers). The increased cost (net) of each of these is entered.

In the area of software, we foresee an increase in the licensing fee of the SCP component, but there is an expected decrease in the fees associated with other program products. In the example, this is due to the consoldation of two smaller systems into a significantly larger complex. As a result, there will be a sizable decrease in per CPU license costs.

With regard to personnel expenditures, we anticipate that some additional costs will be incurred. This value is the result of netting the required manhour costs to accomplish the hardware and software conversions with the reduction in staff that is anticipated due to the merging of two complexes into one and with educating our in-house staff on the new environment (operator training, systems programming internals courses, etc.).

Both on-going and one-time expenses are expected for the physical site. Some up-front preperation work must be completed and additional chilled water capacity is required to support the new equipment. Both of these are reflected in the one-time cost column. For on-going costs, the analyst should obtain the square foot cost used in the standard cost allocation schemes in the company. The required floor space per complex can be obtained from hardware specification manuals or from the vendor. Remember to allow for door clearances and other required extra footage. (The operator has to sit somewhere). Kwh (kilowatt hour power) costs and cooling costs sholud also be obtained from the corporate cost accountants. If these numbers are not available just leave them out and place a footnote in your presentation that denotes the expected percentage difference in costs based on relative space and kwh requirements.

The supplies listed are a one-time purchase of cartridge tapes to be used on the new tape drives.

The overhead values include estimates of normal business expenses (added insurance, personnel benifits, etc.) that are associated with the above items. This value was derived from an overhead formula that was supplied by the corporate accounting staff.

With these values in place, the spreadsheet will automatically fill itself in and yield yearly and total costs. Also, for those who are interested, the net present value of the costs (current year cost column) is supplied. It is interesting to note that the CPU delta of 2 million dollars can grow rapidly (more than twice) when the sundry costs are added and amortized.

The analyst can produce these spreadsheets for several configuration or financial scenarios. It becomes an easy task to point out the relative costs of each alternative. To complete the picture, he needs only discuss the offsetting benefits of the anticipated change. (How to address that topic is another paper).

I believe that if capacity planners consider the financial issues and include them in recommendations the generated proposals will have greater credibility with management and as a result the plan will be more readily accepted and implemented.

General References

[1] Abowd, A., "Understanding Financial Issues in Computer Planning", CMG Proceedings, 1984.

[2] Amembal, S.P., "The Dynamics of Hardware Leasing", CMG Proceedings, 1986.

[3] Anonymous, "How to Buy Enough Computers", Business Week, Decemeber 8, 1980, pp.83-84.

[4] _____, "The Economic Value of Rapid Response Time", IBM Publication, No. GE 20-0752.

[5] Cortada, J.W., "Managing DP Hardware: Capacity Planning, Cost Justification, Availability, and Energy Management", Prentice Hall, Inc., Englewood Cliffs, N.J., 1983.

[6] Febish, G.J., "Data Processing: A Corporate Asset or Liability", CMG Proceedings, 1983.

[7] Ives, B., and M.H. Olson, "Manager or Technician? The Nature of the Information System Manager's Job", MIS Quarterly, Vol. 5, No. 4, December 1981, pp. 49-62.

[8] Morino, M.M., "Establishing an IS Management System", CMG Proceedings, 1986.

[9] Nanjo, H., R. Choy, and C. Liu, "An Information System Cost Analysis Approach", CMG Proceedings, 1984.

[10] Persson, J., "VM Capacity Planning: A Management Perspective", CMG Proceedings, 1983.

[11] Rosenberg, J.L. and E.M. Friedman, "Do I Really Need a 3090 System ?", CMG Proceedings, 1986.

[12] Stanton, J.A., "Managing MVS Installations as a Business", CMG Proceedings, 1984.

[13] Tartaglia, D., "Capacity Planning: Justifying the Need", CMG Proceedings, 1986.

[14] Von Kinsbergen, J.E., "An Integrated System for Capacity Mangement", Journal of Capacity Management, Institute for Software Engineering, Vol. 1, No. 4, 1983.

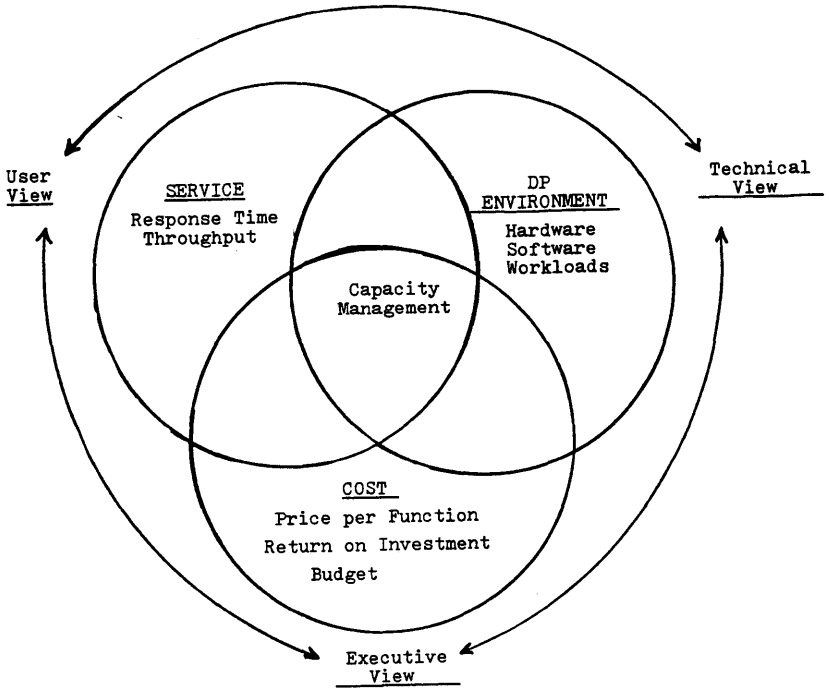


FIGURE 1

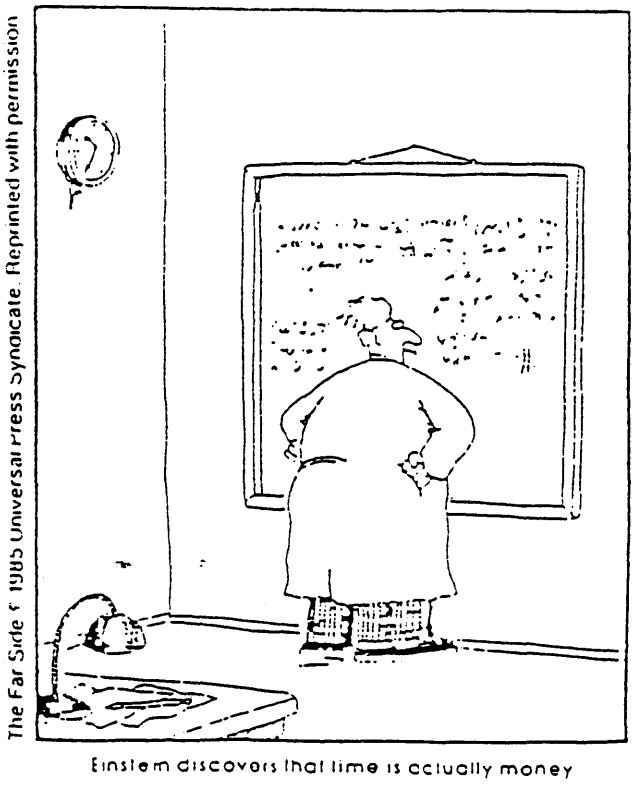

The Far Side © 1985 Universal Press Syndicate. Reprinted with permission
Einstein discovers that time is actually money

FIGURE 2

ASSUMPTIONS

Useful Life (yrs)	4
Inflation Rate	0.1

Cost	Price Delta	One Time Cost	Yr 1	Yr 2	Yr 3	Yr 4	Total Cost	Current Yr Cost
Hardware								
CPU	2,000,000		500,000	550,000	605,000	665,500	2,320,500	2,000,000
Dasd	400,000		100,000	110,000	121,000	133,100	464,100	400,000
Tape	210,000		52,500	57,750	63,525	69,878	243,653	210,000
Other	100,000		25,000	27,500	30,250	33,275	116,025	100,000
Software								
Oper. Sysy	120,000		30,000	33,000	36,300	39,930	139,230	120,000
Sub-sys	(325,000)		(81,250)	(89,375)	(98,313)	(108,144)	(377,081)	(325,000)
Other	(80,000)		(20,000)	(22,000)	(24,200)	(26,620)	(92,820)	(80,000)
People								
Additional	200,000		50,000	55,000	60,500	66,550	232,050	200,000
Education		85,000	85,000	0	0	0	85,000	85,000
Facility								
Space	225,000	220,000	276,250	61,875	68,063	74,869	481,056	445,000
KWH	150,000		37,500	41,250	45,375	49,913	174,038	150,000
Cooling	195,000	150,000	198,750	53,625	58,988	64,886	376,249	345,000
Supplies		85,000	85,000	0	0	0	85,000	85,000
Overhead	100,000		25,000	27,500	30,250	33,275	116,025	100,000
TOTALS	3,295,000	540,000	1,363,750	906,125	996,738	1,096,411	4,363,024	3,835,000

FIGURE 3

CAPACITY BUDGETS AND FINANCIAL BUDGETS: RELATING THEM TO EACH OTHER THROUGH RESOURCE ACCOUNTING

Joseph W. Richardson
J. P. Morgan, Inc.

- Abstract -

Cost acounting can help capacity planners communicate more effectively with corporate management. Although data center chargeout has often been justified, an understanding of both capacity and financial plannng is needed to develop the right system. Deciding what to charge for requires an understanding of computer measurement, so that metrics will be selected that are suited to the reasons why management want a chargeback system. Transforming those metrics into terms that both non-technical and technical management can understand requires an understanding of internal financial analysis - that is, cost accounting.

Resource accounting systems are particularly appropriate vehicles for expressing both computer and financial measurements simultaneously. In additon, they offer the means to charge for reserve capacity (also called "excess capacity") in a way that is perceived as fair by everyone.

This paper presents measurement issues from the perspective of corporate management, and describes the application of both financial and capacity planning techniques. Whenever possible, both techniques are applied to the same issue. Standard rates are described as a means for combining both measurement techniques. Examples are presented which incorporate these ideas. Finally, a combination of financial and capacity planning techniques is recommended as the way to provide corporate management with information needed for control, while providing technical management with accurate, yet meaningful, input to corporate decision making.

INTRODUCTION

Accounting for the use of data centers is not a new subject. Commercial service bureaus have been doing it for years, in order to remain profitable. Many other organizations have also used chargeback techniques for years. Some have simply wanted data center users to be aware that the service is not free, while encouraging them to use the facilities. Others have wanted to allocate the expenses of operating a data center to the company's lines of business (such as the cash dispensing machines provided by many banks). Still other companies want to encourage their end users to choose off-peak hours, or at least to schedule their work so that resource-intensive work will not be submitted at the busiest times. (Development data centers would often like to do this.)

Cost accounting, also called "management accounting," has been used for years by many different kinds of organizations to help management accomplish precisely these goals. Therefore, when problems arise while applying chargeback techniques to data center operations, the cause is usually a lack of common understanding between specialists in financial measurement and computer measurement. If the problem is lack of familiarity with project management techniques, these have been discusssed by Yourdon and others. If the problem is lack of understanding of chargeback issues, these are discussed elsewhere (see "References").

This paper presents measurement issues from the perspective of corporate management, and describes the application of both financial and capacity planning techniques. Whenever possible, both techniques are applied to the same issue. Standard rates are described as a means for combining both measurement ideas. Finally, a combination of financial and capacity planning techniques is recommended as the way to provide corporate management with information necessary for control, while providing technical management with accurate, yet meaningful, input to corporate decision making.

ISSUES FOR DATA CENTER PLANNING

Both capacity planning and financial planning are techniques for corporate planning. Both are used by organizational units to communicate goals and resource requirements to upper corporate management. Among the issues shared by both methodologies when preparing data center plans for the coming year are:

1. How much processing capacity do we need?
2. How much online storage capacity do we need?
3. How much transmission capacity do we need?
4. How much local output capacity (i.e., print or punch) do we need?
5. How far into the future should we plan?
6. Which resources can be shared, and which ones must be dedicated to individual data center users or workloads?

Although the issues are similar, one methodology (capacity planning) usually focuses upon manufacturing capacity (1), while the other (financial planning) usually focuses upon the cost to the corporation. Manufacturing companies are accustomed to combining the two processes. Several other industries are not yet accustomed to this.

The six issues are listed again, but this time typical measurements used by each discipline are also included.

1. How much processing capacity do we have? do we need?
   - Capacity measures include percentage used of total CPU hours available (90th percentile, mean, and average), time available for application programs after determining and applying the capture ratio for the entire system, etc.

(1) if data centers are viewed as factories

Both historical data and input from corporate business plans are required if these or other measures are to be accurate.

- Financial measures include the annual or monthly cost of processors, the cost for work actually done on the processors (assuming that rates do not change during the year), the difference between the monthly cost of providing processors and the monthly charges for work performed by the processors, etc.

  Simple graphs showing resource requirements or expenses (using constant unit charges) over a period of time may be prepared from each set of measurements. Such graphs are frequently helpful when describing how demand has changed over time.

2. How much online storage capacity do we have ? do we need?

   - Capacity measures include cylinders and gigabytes. Analysis may focus on permanent storage plus two uses of temporary storage: temporary datasets (the "passed" datasets of IBM's OS asnd its successors), and permanent datasets which are routinely created and deleted during a normal day's processing (for instance, to enable a steam of jobs to be restarted somewhere in the middle).

     Analysis of VTOCs, VSAM catalogs, and RMF/SMF data provides the material for such studies. Input from people familiar with future business volumes is also required. Abends due to the unavailability of DASD space may provide a means for monitoring the accuracy of the forecasts, although such abends often signal a need to change the scheduling of batch jobs or make other performance adjustments.

   - Financial measures include the cost of the online storage capacity, the cost of that portion which is not allocated to datasets, and the cost of that portion which is allocated to datasets. Financial measures also include the cost of rerunning jobs which abend, because there is not enough space for temporary datasets or for those permanent datasets which are routinely created and deleted each day.

     Both financial measures and hardware measures are especially informative if they are summarized by a definition of "workload" which is meaningful to non-data processing management. If separate chargeback invoices are prepared for each major application or low level end user group, then the invoices may offer a helpful definition of "workload" for planning purposes.

3. How much transmission capacity do we have? do we need?

   - Capacity measures include how often lines or line groups are at least fifty percent occupied during a specified interval, actual network response time (measured in various ways but excluding host response time) compared to desired response time, the frequency with which a receiving node halts transmission to itself, buffer shortages in either an INN or a BNN, etc.

   - Financial measures include the cost of supplying transmission facilities, and late charges caused by busy transmission facilities. Financial measures may also include costs allocated to specific users, if this is technically feasible and can be done economically.

4. How much local output capacity do we have? do we need?

   - Capacity measures include defining local print (or punch) workloads, analyzing the quantity of printed lines or pages per workload, noting the total quantity of printed lines or pages at times of the day/week/month, identifying delays in the availability of printed output which are caused by "printer busy," etc.

   - Financial measures include cost per standard quantity of lines, cost per standard quantity of printed pages, late charges due to the print pool's being overloaded, invoiced costs charged to each workload (if workloads are used for chargeback purposes), etc.

   - Both capacity and financial measures may demonstrate inherent cycles in the workloads when plotted over time.

5. How much additional capacity of each kind do we need as a reserve for the future? How far into the future?

   - Capacity planning techniques include models, as well as trend analyses of cpu / DASD / communication / local print usage, adjusted to reflect next year's business plans. Demand may be forecast in terms of units of total resource consumption, numbers of transactions during peak periods, frequency of peak periods, etc. for the entire data center, major workloads, or both. Recommendations to obtain additional capacity assume that existing capacity is already being properly used by application of performance tuning techniques. These recommendations also assume that, as a result of purchasing or leasing additional equipment, reserve capacity for the data center to grow into will become available in diminishing quantities until additional capacity is obtained.

   - Financial planning measures include similar measurements expressed in dollars, rather than in units of resource consumption or percentage of "busyness". Financial budgeting is the principal technique used to express these forecasts in financial terms. Note that in data center planning, the determination of unit rates as part of a resource accounting cost allocation system is a good example of combining capacity planning results with financial planning results. (This is discussed more fully below.)

6. How will these different kinds of capacity be made available? Will workloads be rescheduled to fit within existing capacity? Will sufficient capacity be purchased to accommodate peak demand without having to reschedule work? If a compromise is necessary, how will it be decided?

   - Capacity measures include availability of resources during peak periods, frequency of peak periods, temporary or permanent use of resources by one or another workload (where a "workload" may be characterized by resource consumption characteristics or operational characteristics), etc.

   - Financial measures include the dollars budgeted for the coming time period (usually one year) by workload, the total expected expenditures which wil be recovered by a chargeback process, the percentage of total expected expenditures which will be deferred for future recovery, etc.

COMBINING CAPACITY PLANNING AND FINANCIAL PLANNING

Effective communication with non-technical management requires that the results of these two planning techniques be combined. Capacity planning presents recommendations in quantitative, but non-financial, terms. Financial planning presents recommendations in financial terms with which corporate management is familiar. However, financial planning alone will not tell management what it really costs to support each workload in the data center. To do that, measurements created by capacity planning must be included.

Resource accounting offers a method for combining the results of both kinds of planning. It gives detailed information about the cost of providing data center resources for all work in which management is interested. Unit rates, if properly constructed, combine the results of both planning methods. When the unit rate for a resource (cpu time, for instance) is computed by dividing estimated total annual consumption into estimated annual cost of the resource, the rate will be a function of both planning techniques. If these unit rates are then used to estimate the yearly cost of each workload in the data center, capacity planning recommendations will have been translated into the cost of supporting each workload in which corporate management is interested.

If resource accounting is the basis for data center chargeback, unit rates can include the cost of reserve capacity (also called "excess capacity") or not, as management desires. Charging for the reserve capacity, which is really a reservior, has been one of the thornier issues. Often, the real problem is telling management how much reserve capacity there is, without requiring corporate management to become engineers. Capacity planning can state how much of each available resource will actually be used (e.g., sixty percent). Management can then decide whether to prorate the real cost to the company between "used" and "reserve," and include only what is "used" in the rates. Naturally, if management decides to do this, the next question is, "When will the deferred charges be included in the rates?" Guidelines for timing the recovery of deferred charges are beyond the scope of this paper. However, resource accounting clarifies the issues, and can improve communication between data center management and corporate management.

Examples of how this method has been used successfully follow. At the end, recommendations for using this method are summarized.

EXAMPLES OF RESOURCE ACCOUNTING FOR DATA CENTERS

1. A cost allocation system for a traditional production environment

   Management wants this chargeback system to allocate the expenses of operating the data center to services marketed by this financial services company. The services could not exist without the data center. Management needs to know what it costs to produce these services, in order to measure their profitability. Management also wants to compare actual and budgeted expenses, to make financial planning as accurate as possible.

   Each fall, during the corporate budget process, resource consumption estimates are prepared. Earlier, questionnaires were sent to the end user areas, and meetings were arranged to collect their business forecasts for the coming year. These forecasts are analyzed together with trends derived from data stored in performance data bases. Discrepancies are reviewed with end user areas. The forecasts are eventually expressed as quantities of various resources that will be consumed during the coming year. Recommendations are made concerning the quantity of each resource that should be available, to ensure sufficient capcity. Both chargeable and non-chargeable resources are included. Chargeable resources are those which appear on invoices (such as DASD). Non-chargeable resources are those which are not charged out separately (such as channels), but which are clearly required for the data center to provide the expected level of service. This stage of the planning process provides information for performance/capacity management, as well as for financial management.

   All expenses associated with the data center (salaries, equipment leases, depreciation, etc.) are identified and assembled. Management decides what portion of these expenses will not be factored into the rates. For example, if a new data center were built which won't be fully occupied for five years, or if a two-year supply of DASD were purchased at discount, some of the actual expenses would be deferred for future recovery. In both of these examples, reserve or "excess" capacity has been acquired. Excluding the cost of reserve capacity is a straightforward way to avoid charging end users for capacity which they cannot use. As that reserve capacity is used, its costs may be included in the rates.

   Finally, unit rates are computed. The expenses which will be factored into the rates are allocated to "resource pools." Each chargeable resource is a resource pool. Allocation is accomplished by the usual cost accounting technique of first allocating direct expenses (such as operators' salaries), then prorating indirect expenses (such as supervisors' salaries) according to direct expenses already allocated.

   Note that the costs of non-chargeable resources, whose identity is important only to performance/capacity management, are allocated to chargeable resources. The intent is to use unit rates to

recover the entire data center budget, except for that part which is deferred. The rate for each chargeable resource is computed by dividing estimated annual consumption into the dollars allocated to that resource. The resulting rate is a "standard cost" which combines the results of capacity planning with financial planning.

These rates are then used to prepare budgets for various workloads. During the year, actual charges are computed using these same unit rates. Actual charges are compared to budgets to provide management information.

2. A cost allocation system used with a pricing system for an information center

Management has a different philosophy for the information center than it has for the traditional production data center. The production center supports applications which constitute part of the company's product line. They are too critical for management to permit them to be run anywhere else but in the corporation's own data center. Management is concerned about the expenses of operating the production data center, but not about whether the production center's applications should be run there at all.

The information center provides data center services which are necessary to the corporation, but which don't necessarily have to be provided by the company's own data centers. Most of the applications on the information center require a few weeks to develop, rather than several months. They tend to require less DASD and other resources. They are usually programmed in FOCUS or a similar language, rather than in a lower-level language. They are almost always oriented towards answering inquiries, instead of towards updating large files with great quantities of data.

These applications could be developed and run using commercial service bureaus. However, the information center is more readily available than outside service bureaus. Since its end users can use either the informaton center or an external resource, information center management keeps its rates competitive with external rates. If the information center's charges are higher than an outside service bureau's charges would be, then end users can decide whether the information center's service is sufficiently better to be worth the higher price.

This information center is not a profit center. Ths is, the information center is not required to show a profit. Its managment is required to demonstrate that it is providing a cost-effective service. To do that, information center management has selected a combination of a pricing system and a cost allocation system. The pricing system is quite detailed, so that end users pay only for those services which they actually use. Whereas the production data center allocates the costs of using five basic resources (CPU time, printers, dedicated DASD space, archive tapes, and specialized communication equipment), the information center charges for ten: CPU time, primary dedicated DASD space, additional dedicated DASD space, printers, special forms, archive tapes, tape mounts, instruction in one or another of the programming languages, programming for an end user, and general consulting. Note that the information center does not charge extra for using the center during prime time. That is because the information center tries to be more affordable than an outside service bureau would be.

Management needs to know how much it costs to support each end user, as well as how much each end user should be charged. For this purpose, five cost pools have been selected: CPU time, dedicated DASD space, tape storage, printers, and staff assistance. These are considered to be the critical resources whose availability controls the quality of service which the information center can provide.

For each end user, both pricing and costing records are maintained. Two sets of rates are used simultaneously: unit pricing and unit costing. The unit pricing rates are determined largely by surveying the competition's rates and from constant discussions with end users about their applications, needs, and budgets. (End users' budgets must be approved in advance by corporate management.) The information center can change its pricing rates during the year, but must be prepared to justify the changes. Unit cost rates are determined the same way that the production center's cost rates are determined (see above) as part of the annual budget process. Unit cost rates can only be changed once each year, so that comparisons of variances between actual and budgeted charges will be meaningful.

Since the unit cost rates ("standard costs") are set the same way as they are for the production data center, they also express a combination of financial planning and hardware capacity planning. Creating and using unit rates as part of a resource accounting system allows management to concentrate on managing the service provided by the information center, without wondering what it is costing to provide that service.

3. A cost allocation system with surcharges and discounts added for use with a development testing environment

The chargeback system for the development testing facility reflects a third philosophy, different from either of the two systems described above. Most of the applications being developed in this facility will eventually be operated in the production data center. They are part of this corporation's product line. Management does not want the details of these applications residing outside the company, where they might be inspected by competitors. However, management still wants the development testing facility to be operated efficiently.

To accomplish these goals, the development facility's chargeback system uses unit costs, and does not use prices. However, due to the kind of demand placed on its resources, surcharges and discounts have also been introduced. Six chargeable resources have been selected: CPU time, dedicated DASD space, shared DASD space, tape storage, tape mounts, and printer. To facilitate the use of surcharges and discounts, job and sysout classes have been created corresponding to low, medium, and high resource contention with other jobs.

The critical variables where "contention" is concerned are CPU time, tape mounts, and printed output. Job classes enable control over the use of CPU time and tape mounts via surcharges and discounts. Sysout classes facilitate control over printed output in the same way. SMF and JES exit routines are used to ensure that published rules about the use of job and sysout classes are enforced by the operating system.

A surcharge is applied to excessively large print outputs during prime time. This surcharge can be avoided by directing such output to a sysout class whose writer is not activated until after prime time. Similarly, surcharges are applied whenever a job is submitted during prime time which uses the high priority job class and calls for any tapes. A job submitted during prime time with any job class except the low priority class, which exceeds its alloted CPU time, is likewise surcharged. The surcharge for excessive CPU time or tape mounts can be avoided by submitting such jobs after prime time, or by specifying job classes whose initiators are not activated during prime time. During third shift, and all day on weekends, discounts are in effect. While discounts are in effect, there is no distinction among job or sysout classes.

Although management has decided on different chargeback philosophies for the production center, information center, and development center, unit cost rates are computed the same way for all three chargeback systems. However, additional steps are required in the development center chargeback system to calculate the surcharges and discounts.

1. Surcharges are computed by estimating what it would cost to provide additional processing capacity, or to provide enough operators to mount tapes or load printers, to ensure that 95 percent of all test jobs submitted during prime time would be turned around according to the service agreements, no matter what their resource requirements. The estimated additional expense is then divided by 260 (the standard number of work days in a year), then divided again by 100 (the estimated number of excessive jobs which would be submitted each day if there were no surcharge - see footnote 2). The result is a surcharge per job, whether the surcharge is due to tape mounts, excessive CPU requirements, or excessively large printouts.

2. The discounts are simply negative surcharges. That is, each job which is submitted during the times that discounts are in effect receives a discount equal to what the surcharge would be.

The system of surcharges and discounts was instituted to allow for the unexpected. The development managers' best estimates are factored into normal cost recovery during prime time. However, product development always includes some surprises. The surcharges and discounts make it easier for management to express how much an immediate solution is worth to the company in dollars and cents.

(2) The number of jobs changes annually based on experience.

CONCLUSION

Resource accounting offers a useful way to describe the results of capacity planning in terms which corporate management can understand. The technique, also called "cost accounting," has been used in manufacturing for years. By viewing the data center as containing resources which are consumed by the work being done there, these techniques can readily be applied to data center accounting. Cost accounting, after all, was developed to give management better control over the expenses incurred while producing the goods and services which each company offers for sale.

There have been many discusions about charging for "excess," or reserve capacity. End users resist accounting systems which charge them for resources they do not use. If resource accounting is the basis for data center chargeback, management has the opportunity to defer charging for resources until they are used. By developing and using "standard rates" to compute end users' charges, the problem of charging for "excess capacity" can be resolved.

REFERENCES

1. "Stages of Growth in Chargeback Systems," J.W. Richardson, CMG '86 Conference Proceedings.

6027S

EXPERIENCE IN COST ALLOCATION

Joseph Stancavish
Manufacturers Hanover

ABSTRACT

Cost allocation is often referred to as an art and not a science. Those responsible for allocating costs would like cost allocation to be as exact as possible in order to defend any costs that users of data center services receive. Where measurements are not exact, a judicious combination of data center performance experience along with fimiliarity with data center operations is needed to satisfactorily resolve unclear issues. This paper addresses the author's experience in cost allocation at a large money center bank.

DESCRIPTION OF THE CURRENT ENVIRONMENT

Manufacturers Hanover Corporation is a large New York based money center bank providing a multitude of banking services for wholesale and retail customers. Many data centers support these services. The data center that is the object of this paper is called Wholesale Computing Facility (WCF). As its name implies, WCF supports service to wholesale customers of Manufacturers Hanover. An example of a wholesale customer is another bank or corporation. The range of services provided include electronic banking, check reconcilement, real estate and mortgage, letter of credit, domestic and international payments, commericial loans, demand deposits and investigations. Each of these applications has corresponding products for which costs must be calculated.

The WCF has a 3084 QX processor along with a 3090-200 processor. MVS/XA with JES3, VTAM/TCAM, CICS and IMS are the major software products in use. The 3084 is normally the production system with test on the 3090. There are approximately 250 gigabytes of DASD.

ACCOUNTING STRUCTURE

A product is defined as a result of data processing work. An example of a bank product is a check reconcilement. Each product that uses WCF resources is charged by measuring resource usage in a number of areas. These areas include CPU usage, DASD storage, print usage, tape storage and mounts, contingency data center service emergency maintainance and others. These areas are called activities. Costs are then assigned to each activity in the data center. These costs are a combination of equipment, salary, occupancy, etc. From the assignment of costs to activities, a unit cost per item of activity can be calculated. When each product's resource usage is measured a total charge can be calculated.

An analogous situation is an automobile assembly line. The product is a car. In order to produce a car, what are the cost involved? First, activities must be defined. The activities could be: assembling the body, inserting the engine, installing the interior, painting the car, testing it, etc. Second, what are the costs associated with assembling the body? For each activity one would need to know the costs of materials, labor, occupancy, power, etc. If 50 car bodies were assembled in one day with an associated cost per day, one could calculate the cost per assembly. This would have to be done for each activity. The result would be a total cost per car. One could then price a car accordingly on the market to recover the costs and perhaps make a profit.

Is it that easy to charge for data processing services? Of course not!

PRACTICAL CONSIDERATIONS

The steps to an accurate and equitable cost allocation system are:

1. Develop accurate and complete inventories of all resources being charged.
2. Have a database of costs for each resource being charged.
3. Relate resources to products on a one to one basis where possible.

Developing accurate and complete inventories for all resources (e.g. CPUs, DASD, staff) is necessary to obtain the costs of data center services. The costs are obtained by looking at invoices associated with the items on the inventories. Relating resources to products one a one to one basis if difficult because of the sharing of respurces in the data center. Wherever possible the one to one relating of products to resources should be done because it simplifies the job considerably. More importantly, it reduces any overhead allocation issues that arise.

CURRENT IMPLEMENTATION

The major components of the cost allocation system are: a data collection program to gather resource usage on the desired activity and assign the usage to an appropriateproduct and a program to calculate the charges for distribution to the appropriate area.

The current implementation of the cost allocation system has various sources of input: SMF 30 records, SMF 6 records, the tape master catalog, a DASD database and manual input. The SMF type 30 subtype 5 record is used to record data on the CPU and tape mounts. The SMF type 6 record is used to record estimated data on print pages. The tape master catalog (TMC) is the source of tape storage information. The DASD database supplies data on storage used by applications. The manual input is for activities like contingency and emergency maintenance. LANDMARK monitor data is used for CICS regions.

The data collector portion is written in SAS. A combination of tables and code are used to assign resources to particular products. Job name or TSO userid is the indentifier for CPU, print, tape storage and tape mount activity. Dataset names and corresponding levels are used for the assignment of DASD to products. For example, in the dataset A.B.C: A is level one, B is level two and C is level three. In the case of CICS regions, the transactions ID is the identifiers are used for other activities. An example of this is a table of port assignments for telecommunications ports.

An example of the data collector output is shown in Table 1. Across the page lie a subset of the total activities measured by the data center. Along the left side are the line items. These can be products, product groups, CICS regions, overheads and test departments. The ideal situation is to have the data collector show only product usage with no ambiguilty due to shared resources. This ideal is only partially acheived. Product groups are shown where there is some ambiguilty as to particular product assignment but where the general area of assignment is known.

Overheads are shown when there are resources used for the benefir of the data center as a whole and not any particular product or product group. CICS regions are shown as the whole region. Another program must analyze the CICS data to assign it to the appropriate product. Test departments have a table of TSO users assigned to them. Any charges produced by the test departments are then charged to the product areas by the test department themselves.

1. Distribute unused capacity according to actual usuage.
2. Distribute unused capacity equally to all products.

The first method is the one used in the current implementation. The second method is unfair to products which have small actual usage. The product may have more unused capacity charges than actual use charges.

TABLE 1. OUTPUT OF DATA COLLECTOR.

	CPU		PRINT		TAPES STORED		PRIVATE MOUNTS		SCRATCH MOUNTS		DASD SPACE	
DESCRIPTION	PCTSUM	SUM	PCTSUM	SUM	PCTSUM	SUM	PCTSUM	SUM	PCTSUM	SUM	PCTSUM	SUM
CHECK-RECONCILEMENT	14.22	32	3.36	110	23.43	97	6.51	37	36.20	311	3.96	200.00
CHIPS-GROUP	9.33	21	1.19	39	4.83	20	69.01	392	27.59	237	5.58	282.00
CICS-REGION-CLS	22.22	50	2.90	95	11.35	47	6.34	36	11.41	98	0.16	8.00
DEPARTMENT-5746	6.22	14	0.06	2	1.93	8	1.23	7	0.58	5	1.76	89.00
GLOBAL-EXPOSURE-SYSTEM	7.56	17	85.94	2812	7.00	29	3.35	19	0.00	0	57.17	2888.00
INTERNATIONAL-INVESTIGATIONS	14.67	33	1.74	57	5.31	22	3.87	22	11.53	99	20.78	1050.00
ONLINE-OVERHEAD	6.67	15	2.84	93	19.32	80	3.52	20	7.45	64	0.55	28.00
SYSTEM-OVERHEAD	9.78	22	0.37	12	12.08	50	3.35	19	0.00	0	9.74	492.00
SYSTEM-SUPPORT	9.33	21	1.59	52	14.73	61	2.82	16	5.24	45	0.30	15.00
ALL	100.00	225	100.00	3272	100.00	414	100.00	568	100.00	859	100.00	5052.00

All this data was then organized into a spreadsheet. Rules to assign overhead distributions were developed with the help of the controllers department. The resulting activity usage was then input to another system to distribute expenses.

CAPACITY CONSIDERATIONS

An interesting problem arises in this system. This methos of cost allocation is called a full absorption system in the literture. Full absorption means that all costs must be recovered. However, full cost recovery means that resources are being paid for even though they may be unused. This leads to the interesting question: who pays for unused resources?

The CPU activity is particularly interesting. Where production online systems are involved, the CPU must be sized sufficiently large to handle any peaks. The resulting configuration handles online peaks well but leaves non-prime time with an excess of capacity. Add a backup processor, and an already large cost has doubled. What is evident here is that the online region peaks are driving the size of the processor up. The online regions, should therefore, pay the portion of the CPU which was needed to handle their requirements, In the current implementation this translates into a surcharge for use of an online region. The surcharge can be handled in a number of ways: a flat charge for using a region, an increased unit cost for online CPU usuage, increased activity volume for use of an online region with the same unit cost as batch. The current implementation utilizes increased activity in addition to the actual usage of each region to calculate the surcharge for users of online regions.

For other activities there are two basic methods to distribute unused capacity:

COST DEVELOPMENT

Activity volume was estimated for the coming year based on the activity volume of the previous year. Table 2 shows an assignment of costs to the various pools. Note that activiry volume should include explicitly some allowance for unused capacity for each activity pool. The result of dividing the total dollars in each pool by the total activity volume for that pool is the unit cost for a unit of activity in each pool. If there is a mapping between product transactions and the various units of activity a cost per transaction can calculated. User estimates of the product volumes could then be used to charge each user area.

Table 2. Allocation of costs to various activities.

	CPU	PRINT	TAPE MOUNTS	DASD	CONTINGENCY
SALARIES	$1,000,000	$200,000	$100,000	$500,000	$300,000
BENEFITS	$200,000	$50,000	$25,000	$30,000	$35,000
EQUIPMENT	$3,000,000	$100,000	$30,000	$900,000	$500,000
DEPRECIATION	$1,000,000	$50,000	$10,000	$300,000	$200,000
OCCUPANCY	$500,000	$20,000	$10,000	$200,000	$100,000
TOTAL	$5,700,000	$420,000	$175,000	$1,930,000	$1,135,000

DATA CENTER PERSPECTIVE

Because many users have difficulty defining what their products are and as a result have difficulty estimating product volume, there is a viewpoint which tries to remove some of the complexity of the cost allocation process. That idea does not require the end user to estimate his product volume but only to look at his old "bills" and see what he has used previously. He can state his expected actual usuage for the coming year. He doesn't have to know the relationshio between data processing resources and his products. Product volume estimates are not reliable.

FURTHER DEVELOPMENTS

The author is currently involved in the development of a PC based cost allocation system. The basic idea is to have the mainframe do the heavy processing for download to a PC. A PC database package can then be used to produce bills, simulate the effects of cost changes and debit or credit departments as necessary. The power of the PC based system is that changes to charges can be made quickly and easily.

CHARGEBACK VS. COST ACCOUNTING
THE APPROPRIATE EMPHASIS

George J. Gillen, Jr. and Billy F. Henry
Coopers & Lybrand

ABSTRACT

During the past five years, we have worked with several clients throughout the country helping with the implementation of Information Services Cost Accounting or "Chargeback" systems. This paper addresses the issues that have surfaced from company executives regarding financial management of Information Services.

More and more, implementation of a system for charging Information Services users is taking a back seat to a more comprehensive and important issue: implementation of a system that encourages effective utilization and management of information system resources.

Our purpose is to provide the reader with insight into the value that an effective Cost Accounting System can provide to both Corporate and Information Services management.

I. THE STATE OF INFORMATION MANAGEMENT

OUR EVOLUTION

From the age of the Abacus, through the era of the Slide Rule and up to the introduction of the Personal System/2*, one issue has pervaded the calculating device arena: "How can I do more, in less time, at a cheaper cost?" As a computer literate society, we will continue to strive for a better mouse trap to enable our organizations to gain a competitive edge while reducing overall costs.

However, one dilemma continues to plague management both within and outside of the Information Services (I/S) organization. The area of concern commanding management's attention more and more is cost. "I need to know what it costs to provide services so I can make intelligent financial decisions".

WHERE ARE WE

We in the I/S profession have done a pretty fair job of providing information technology to our corporate users. Engineers have given us faster processors and storage devices, card reader/sorters are cornerstones in the Smithsonian, and every manager with the desire, can run a Lotus spread sheet and access this mornings sales statistics from the comfort of his/her office or even home.

Technologically, we have progressed further and faster than any other area of the business. From a management standpoint, however, far too many of us rate a C or worse. Much to our chagrin, technology has not managed itself and we have not taken the steps necessary to manage it. It is our belief that those who will survive over the course of the next decade will be those who actively alter this course and aggressively manage the I/S business unit.

THE CORPORATE MANAGEMENT PERSPECTIVE

Given this as the State of Information Services today throughout much of corporate America, what does company management think of us? Below are some of the feelings expressed to us by company executives we have worked with over the past five years:

.. DP is still a black hole where corporate funds go in and very little comes out

.. Nobody in that group can tell me what it costs to run our application portfolio

.. If new technology is cheaper, why does the DP budget keep going up

.. The answer from that group is always more hardware, new software, more people; in short, more money

.. I would really like to find a way to measure my ROI for DP.

Two themes are consistent in these viewpoints. First, they keep referring to the "DP" department. Obviously we have not demonstrated to company management that our job is to manage information for the corporation. We are, or at least should be, more than just processors of data. Machines do a better job of that anyway.

Secondly, management seems overly concerned with money. In point of fact, they had better be or their successors surely will be. Today's environment demands that we be as cost conscious as possible. We need to ensure that every department is making the most out of every dollar spent and that the organization is functioning as efficiently as possible.

The I/S team is no different. We have the capability to financially manage and report on our activities just as any other department does. In fact, we can probably do a better job than most because of the tools available to us, and in most cases our vast experience.

II. COST ACCOUNTING VS. CHARGEBACK

Perhaps the best management tool available to us is a Cost Accounting system for I/S. We are talking here about a system that goes far beyond the traditional chargeback system many of us are familiar with.

WHAT'S THE DIFFERENCE

Most pure chargeback systems have one major objective: to identify aggregate costs and allocate those costs back to the user community such that, by month end or year end, the I/S budget has been accounted for.

The major problem we have seen with most of these systems is that they provide little or nothing in the way of management information. Charges are often not reflective of resources consumed. Changes in user behavior have little or no effect on their bills. Many services are deemed free by the user community resulting in costly and inefficient use (thereby driving I/S budgets up). Budget increases can only be justified in terms of nebulous increasing user demand. And many users end up paying for someone else. The result is mistrust and skepticism on the part of the users and management.

Enter the I/S Cost Accounting system. While using much of the raw data and statistics used for chargeback, the goals and objectives are very different. Fundamentally, we are proposing a management information system. And

by the way, we can charge users for services provided as well.

The basic difference in the two approaches is that a chargeback system strives to spread the I/S budget among the user community in some manner. If we also get some management reporting, so much the better. With an I/S cost accounting system, however, the primary goal is to identify costs. It allows you to monitor and manage such things as:

.. Budgeted vs. Actual Recovery
.. Budgeted vs. Actual Resource Consumption
.. Rate Performance
.. Budgeted vs. Actual Expenses
.. Year to Date Performance
.. Year End Forecasts/Reforecasts

The bottom line is that we now have a tool to help us manage costs. Costs which can very easily be allocated back to the user community fairly, accurately, and consistently.

SYSTEM OBJECTIVES

As we mentioned above, our objective in implementing a cost accounting system is to help us better manage the I/S function. The key objectives of the system should be:

.. The identification and accumulation of costs by natural cost category (i.e., Cost Pool)

.. The integration of cost accounting functions and other appropriate I/S management systems such as project and time management, capacity planning and telecommunications management

.. An I/S modeling capability for analyzing cost relationships, forecasting the cost of alternative service levels, and projecting standard service rates for budget and chargeback purposes

.. The billing of user organizations based on actual utilization of resources and standard rates.

Our goal is to run the I/S function as an integral part of the overall business organization regardless of whether your charter is service bureau, cost center, or profit center oriented. The cost accounting system is the tool that can help us.

CRITICAL SUCCESS FACTORS

Successful implementation of a sound I/S Cost Accounting system is not as easy as selecting and installing a software package, populating some tables and running the package against SMF tapes. There are several factors critical to the success of the project.

First, is Management Commitment. The projects we have seen enjoy the biggest success are those with the support of senior management throughout the company. More is required than just lip service.

Secondly, management must be involved. During the course of the implementation, several wide ranging decisions will have to be made, some requiring senior management approval. Keep senior management aware of the project. If they are kept apprized of the issues, they are more likely to be willing to make the tough decisions and support your efforts.

Third, no project of this size and complexity can be successful without sound project planning. Time spent up front laying out the detailed course of action to be followed may well be the key to a successful implementation. Several technical and functional areas will be involved throughout the project and their participation must be planned for. An effort of several weeks up front can save several headaches down the road.

The fourth critical success factor is Investment. This project will take time, effort, and money.

The amount of time you spend could range from 4 or 5 months to almost two years. This is dependent on how large the environment is (multiple large data centers or a one CPU site) and the makeup of the project team (the more experience on the team the shorter the time frame). Also, the use of external consulting assistance with directly applicable experience can significantly reduce the overall time needed.

To be successful in a reasonable amount of time, a great deal of effort must be expended. It is likely that a team of 3 to 5 full time people will be needed, supplemented throughout the project by specific areas of expertise such as Capacity Planning, Network Management, Applications Programming, and so on.

As we all know, there are no free lunches. This project will cost you money. The cost elements involved are; software package(s), outside assistance, and the cost of your own people. On the average, the total project cost should be around 2% to 3% of the I/S budget. When you think about it, that is a small price to pay to be able to have the system in place to help you manage that large (and ever growing) I/S budget. You might also consider that this is a one time charge that, with on-going care and feeding, provides a long term management solution.

TO CHARGE OR NOT TO CHARGE

The decision to actually invoice users for the cost of service provided can only be made within the business and political framework of your own organization. One of the largest impediments to charging is the issue of whether users can, in fact, have some control over the costs associated with I/S. Skepticism regarding the chargeback system and process is rampant in the industry and often with good reason. If the issue is approached through implementation of a true cost accounting system coupled with user involvement and education, the results can be surprising. Few people object to paying a fair price for good service and I/S services are no different. If a user feels comfortable with his charges (i.e., he understands what they are) and sees that different behavior on his part will affect his costs, the battle has been won. A true cost accounting system is your best weapon in this fight.

THE NEED FOR MANAGEMENT REPORTING

Central to the issue of more effective management of the I/S organization is the right information, not just "neat technical data". While EXCP counts may impact performance, they don't help us manage user relations any better. Neither will graphs of increasing CICS transactions help us sell management on the need for additional memory. We need to present the user community and corporate management with meaningful information from which intelligent decisions can be made. By demonstrating what resources are being consumed by whom (or by which applications) and presenting real cost figures, we appear to be much more business oriented. And we are. Too often in the past, the cost part of "Cost/Benefit" analysis has been merely a guess, and a poor one at that.

By using information provided by the cost accounting system we can effectively assist our management to understand:

.. What maintenance really costs
.. What development really costs

.. What systems are resource "hogs" and require modification, rewrite or replacement
.. What it costs to support a remote terminal.

In short, good management reporting from a sound cost accounting system can tell management where the I/S budget is going.

WHAT DO I GET?

After all of the blood, sweat, and tears of getting the system implemented, what have I really accomplished?

Hopefully, life will become easier (if that's ever possible in the I/S environment). From the standpoint of managing your own organization, you've developed a management control system that will help you run your organization more efficiently. You'll feel much more comfortable that you understand what's really happening in your department.

Next, you have the ability to give senior management something they've wanted for years: management information; financial data; cost analyses. They can now look at I/S the way they view other organizations and they'll feel more comfortable that you, too, can think as a business manager.

And let us not forget the people who really pay your salary; the users. In every client situation we've experienced, communication between I/S and the user community has opened up. To be successful, the implementation requires user involvement, and this opens the avenues of communication. The common mistrust and skepticism we spoke of earlier begins to fade and is replaced by understanding.

And finally, there is something that the I/S organization, Senior Management and the user community all share in; Defined Expectations. Each group now has an understanding of what is expected from the other.

* Personal System/2 is a trademark of the IBM Corporation.

PERFORMANCE ENGINEERING: SOME LESSONS FROM THE TRENCHES

Dr. T. E. Bell, Rivendel Consultants Inc.
A. M. Falk, TRW Defense Systems Group

ABSTRACT

Performance opportunities and problems exist in any application system. Performance Engineering exploits the opportunities while avoiding the most critical problems. It consists of performance prediction early in development, of tracking design and implementation, and of correcting performance problems throughout system life. Unfortunately, development managers are sometimes unsympathetic, vendor claims are often overly optimistic, and performance drifts away if not continually monitored. We suggest using simple models that are integrated and cover the whole system, making early estimates of resource consumption, taking measurements as early as possible, and tracking performance continually.

The implication from reading much academic and industry literature is that good, properly-motivated developers pursuing a structured development approach do everything right the first time. The authors of that literature live in a much simpler world than we do. Out here in the trenches we live with hard problems, with limited resources, with difficult performance requirements, and with our own mistakes. In this paper we present an overview of Performance Engineering as we have experienced it, a series of specific topics we have found to be important, and a set of implications and conclusions. The two objectives of this paper are 1) to indicate some of the practical lessons we have learned about doing Performance Engineering, and 2) to provide confirmation that Performance Engineering can yield great pay-offs to software development projects.

In addressing these objectives, we use examples from several systems whose characteristics are disguised. We must disguise the situations because of possible embarrassment to the dedicated developers involved.

PERFORMANCE ENGINEERING

Performance Engineering should be an integral part of developing any large system [1, 2, 3] -- as well as most small ones. But is the effort really the dry exercise in analytic modeling that is implied in much of the literature [4]? Our practical experience indicates that a good deal more is necessary for success. We have needed to work closely with developers so they feel that we're their personal risk-reduction technique rather than their adversary. (When we're really successful, we're confronted by the problem of developers complaining about the inadequate time we can devote to their specific problems.)

If Performance Engineering only encompassed a series of models, primarily based on simulation and queuing theory, it would not be particularly useful. We view Performance Engineering as being a far more fruitful effort -- composed of the projection, tracking, and improvement of application systems. This activity occurs during development, deployment, and operation. It uses a variety of techniques, but complexity of technique is a disadvantage rather than a desirable attribute.

Our approach to Performance Engineering involves making early **projections** and then measuring early performance in order to **track** against those projections. (Simply generating the projections is itself useful because it forces us to concentrate on the issues of particular importance to performance, and to identify the topics that must be worked to ensure that we have a solid baseline.) Usually, we have been shocked by the difference between the projection and the attained performance in early versions of the system. We must then work with developers to **correct** the problems so that the newly-projected performance can be brought to the level of requirements. This is an iterative procedure, but we have found that "corrected" approaches (when created with the active aid of developers) frequently come quite close to the required performance.

We also work with the suppliers of the hardware resources and system software to ensure that the architecture for a dedicated system is appropriate for the application. This turns out to be considerably more difficult than might be anticipated; the range of alternatives is very large, and the reliability of advertising data from vendors is frequently very low. Costs are usually considerable, so our projections must be reasonably accurate -- even though the applications are not yet implemented, and development efforts often go in unanticipated directions.

Performance Engineering is really difficult, and the number of wrong answers is truly impressive. It is also personally interesting, professionally rewarding, and an area of extremely high payoff for organizations developing large software systems. However, success in Performance Engineering requires a good deal more than just building a large simulation model or analytical model. Our experience is that the Performance Engineer needs to cope with a variety of unpleasant realities and to pursue ambiguous situations with simple models. The observations and suggestions listed below arise from our experiences in a number of projects.

UNPLEASANT REALITIES

We have observed several unpleasant realities in software development. Performance Engineers in some organizations may be immune to some of these, but we have observed them in different organizations undertaking a variety of different types of developments. Below, we present some of our observations along with some explanations.

MANY DEVELOPMENT MANAGERS BELIEVE THAT THEIR ROLE IS TO BUILD FUNCTIONALITY -- WITH PERFORMANCE SOMEONE ELSE'S RESPONSIBILITY

First-level development managers frequently believe that performance is the responsibility of someone else. This attitude is probably reinforced by the tendency of users and project managers to define functionality first, obtain an estimate of development costs, and then add performance requirements. However, we have observed instances in which this easy explanation is demonstrably untrue; performance requirements were articulated early, were agreed to by developers, were

recommitted to during the project, and then were viewed as someone else's problem. This situation seems to arise from several sources.

First, developers often find it easier to devote their efforts to finding that next functional software bug than to tackling tougher performance issues. Performance problems are often difficult to solve, require technological knowledge about the system (both hardware and software), and involve sophisticated approaches and tools. A first-level development manager was probably promoted to that level by being able to implement functionality, and he may find the new topics neither an interesting learning experience nor relevant to the world in which he has succeeded. Performance issues are both an intrusion and a threat to his position.

Second, a severe performance problem can mean a substantial redesign effort. Just getting the required functionality is difficult, and satisfying all these "other" requirements may result in a major redevelopment effort. The criticism implied by a demand for redesign (or just the probability of a review of his design) can be an affront to the individual. Many developers regard software development as an art, and their code as an expression of their own personal creativity. They can treat their code so possessively that any intrusion into their domain by a Performance Engineer is greeted with open hostility. One of our favorite comments by a developer is "If you're so smart, why don't you get in here and do some of the hard work?"

Third, developers don't have a natural understanding of Performance Engineering; unless they have seen successes from intensive analysis, they are skeptical. Many view the performance of a system as an uncontrollable result of the development process. In one case, a developer being reviewed argued that everything had been done to improve performance -- even including implementing the whole system in Assembler language. Therefore, examining the system or measuring it was simply an exercise in futility; his product was certainly the very best that could be obtained because of his dedication.

Fourth, redesign (or even minor revision) for the sake of performance can be disruptive and mean schedule slippage and adverse cost impacts. Such changes will be very unwelcome to the development staff who are already under pressure by management to produce the system on-time and on-budget. The easier path is to get something up and running -- and be able to report success. If necessary, those annoying Performance Engineers can be neutralized by simple hostility.

We have observed all the above situations, and have found some difficulty in remaining dispassionate when a developer's defense takes the form of personal attacks. In one case, the development manager's opening gambit was to ask how the Performance Engineering organization was going to fund any disruptions to his people's work. In another case the manager of a critical module wanted to turn over his work to the integration team to find out how the system would function as soon as possible. Then he'd go on to other work and leave the performance issues to someone else -- the Performance Engineering team. If your situation involves limited resources or constrained response time, you may find these tactics objectionable. If not, you're not involved in Performance Engineering.

Project managers can undertake several actions to limit the dysfunctional effects of developers distancing themselves from performance requirements. First, they can have performance requirements (eg, resource budgets) stated prior to the beginning of the project. Second, they can give appropriate weights to functionality and performance in early software deliveries; later deliveries will reflect the relative importance obvious from the manager's earlier actions. Third, in performance-critical situations managers can get personally involved in the issues on occasion. In one successful case, the project manager reviewed important performance memoranda (about two per year) and added his name as an author; the implication from his involvement was clear to the developers.

VENDOR PERFORMANCE STATEMENTS ARE OFTEN NOT USEFUL FOR MAKING PROJECTIONS

The computer marketplace is intensely competitive. Vendors make claims about their system hardware and software performance, and it is the Performance Engineer's responsibility to determine the validity of these claims and their applicability to the project. Aggressive vendors of "fourth-generation" software and alternative hardware products are particularly prone to optimistic generalization of performance figures achieved under narrow conditions. In hardware, for example, the familiar MIPS rating (Meaningless Indicator of Processor Speed) is a good example of the problem. These ratings are often based on running programs such as the Towers of Hanoi or the Whetstone Benchmark and bear little resemblance to any user application. They are typically small enough to be executed completely in the CPU's cache, so processing speed is far higher than can normally be attained. Only the user with the simplest of applications will come close to achieving the speeds implied by the vendor claims.

One of the most inappropriate techniques is to adopt a single indicator of performance (like MIPS) and blindly make projections for a complex environment. For example, one minicomputer vendor advertises a 10 MIPS rating for his machines, an impressive rate for his price -- if his machines were that fast. In fact, we have found that a rating of 3.2 is more appropriate for a variety of realistic applications. Many-faceted benchmarks (eg, the one described in [5]) are far more useful for a competent analysis. The Performance Engineer will be most successful if he provides his own benchmark and has all vendors (of CPUs, tape drives, disk drives, memory, communication front ends, etc) run that specific benchmark. The selection process can be simplified by discarding those vendors who refuse to run the benchmark; any selected system should be reevaluated if performance data are not available.

In the software realm, claims are similarly impressive, and doubtful. Such claims extend from microcomputers up through main-frames. For example, "vapor-ware" (software that has been announced but exists only as an idea -- vapor, and may never be any more firm) is a common problem. Its performance will be really fantastic "any day now". In the main-frame marketplace, performance claims have actually led to legal action on the basis that they are not legitimate. The Performance Engineer should refuse to accept the claims of vendors about performance unless they can be verified by meaningful benchmarking or (if necessary for forthcoming products) by doing an independent projection.

A large law firm purchased an accounting and word processing system from a small start-up firm specializing in software packages for the legal profession. Although the package had not been fully implemented, the selection committee was swayed by the vendor's claim that comparable features and performance would be provided at a much lower price than other packages offered by more-established vendors. The package not only failed to function as advertised, but it ran very slowly. Even though the firm had

competency to seek redress in court, this would hardly solve its needs for a system that performs well; the issue remains unresolved. Of course, the firm would advise a client against accepting a contract without legal review -- even if the other party assured the client that everything was perfectly fine. Yet it accepted the assurances of the software vendor of the appropriateness and speed of its product -- without a performance review.

PERFORMANCE SLIPS AWAY UNLESS IT IS CONTINUALLY MONITORED

Developers, for the most part, are dedicated and over-committed people, even though some of them would really prefer for performance issues to go away. They must make decisions about design and implementation every day, and they can hardly consider every possible issue each moment of the day. As they make their decisions, unexpected performance effects creep into the system, few of them beneficial. Many factors limiting performance are not recognized at the beginning of a development project, but the effects of these must be reflected in revised projections as soon as possible. Simply doing an initial analysis, making a projection, and then leaving things alone doesn't yield very good results.

After implementation has begun, performance of important subsystems must be reevaluated on nearly a continuous basis. With such reevaluation, unfortunate directions can be identified immediately, and corrective actions may be possible before extensive resources are devoted to the wrong option.

When unfortunate performance impacts are not avoidable, projections must be reassessed so that the project can determine whether compensating actions are possible. In one case, a DBMS for a real-time system was divided into several independent tasks to reduce memory impact. The overhead from message passing between tasks for each I/O was so high, however, that the CPU time per I/O doubled, and we projected an increase in system CPU consumption of over 10%. Early measurement identified the problem, and corrections were feasible before personnel were assigned to other activities.

In a second case, our early analysis of an important transaction indicated that it would be within its I/O and CPU budgets. However, as the design progressed, we realized that our analysis was incomplete; the sequence of activities would result in unacceptable response time. Therefore, we suggested that the application be divided into two parts that could be executed in parallel. By tracking performance (even before we could do any measurement), we were able to identify needed design revisions before implementation began.

In yet another case we tracked performance in several subsystems on a large project, and projected that performance would be adequate in one particularly challenging subsystem. Subsequently, we were preoccupied elsewhere and ignored that subsystem. Two months later we reexamined the subsystem (by doing some quick measurements) and found that a variety of problems had arisen. Considerable effort was required to bring performance back under control.

As systems are designed and evolve, performance problems invade and remain. As more effort is invested, the cost of removing the problems goes up.

EFFECTIVE TACTICS

Below, we suggest some tactics to make Performance Engineering more effective. These suggestions do not arise from any abstract theory or hopes for future software development practices. They come from our bitter experiences and sweet successes.

USE SIMPLE MODELS SO THAT FREQUENT CHANGES CAN BE ACCOMMODATED

With a really complete model of system performance, it seems that the Performance Engineer should be able to predict the system's throughput and response time. He should be able to compare predicted performance with achieved performance and identify deviations. He should be able to apply theory throughout the system. In fact, he will spend the bulk of his time detailing the model and trying to modify it to reflect the system's evolution. The effort required to build and maintain a complex model is effort diverted from addressing performance issues important to the system's success.

Instead, a simple model quantifying resource consumption and/or estimating throughput can guide efforts to the most important topics without requiring an excessive development effort just to implement the model. Probably, developing even a simple model will require the Performance Engineer to discuss the design of subsystems with developers and then to suggest the important performance aspects of the designs. Simply going through the process of defining these aspects of the design will provide opportunities for the Performance Engineer to work with developers and identify improvements -- as early as possible. In most cases it will surface unexpected areas of resource inadequacy and performance over-commitment, and it will provide a review of design from a new point of view. Problems can be jointly identified and jointly resolved. Of course improvements will require revisions in the model in order to reflect the new design, but modifying a simple model is straightforward.

In two cases organizations decided to develop extensive discrete-event simulation models, and assigned several people to undertake the effort. The models were developed, but lagged the development of the systems to such an extent that, in one case, a second model development was undertaken so that development trade-offs could be made in time to influence the design of the system. In the other case the complex model was replaced by a simple spread-sheet model because of difficulties in understanding the reasons for some results of the simulation model. Neither the developers nor the modelers could give an explanation for the interactions, so it was difficult to suggest alternatives for system improvement. The simpler model was easier to understand and therefore more useful.

(Models of existing systems are typically used to determine the effects of alternative hardware configurations, or to indicate the capacity limits of the current hardware with the existing software. These models have the advantage of being based on empirical data showing the results of system interactions, so far less speculation is required on the part of the performance analyst.)

MAINTAIN A GLOBAL MODEL REFLECTING SYSTEM-WIDE PERFORMANCE EFFECTS RATHER THAN A SET OF INDEPENDENT SUBSYSTEM MODELS

As design issues arise, quantitative analyses must be executed to support decision-making. As a project progresses, a variety of these analyses will have been done, and they can be used to project system throughput. Unfortunately, inconsistencies tend to creep into these analyses as they are updated on different schedules and with different objectives. The

result can easily be a deceptive set of answers for questions raised late in the project's life.

We strongly recommend maintaining a consistent model of resource consumption across the system. This model should be updated with each major modification of the system and each special performance analysis. The sources of values and relationships should be documented so that later revisions can be made without re-deriving the model.

In one large project we did a variety of performance analyses concentrating on assisting design personnel. The individual analyses sufficed until a major projection was required. At that time we found that, although we had diligently documented each individual analysis, we could not perform the required projection until we resolved a variety of inconsistencies. Resolving these was difficult because of undocumented and inconsistent assumptions implicit in various analyses. For example, a change in hardware resulted from an analysis of one subsystem that used that hardware extensively. The effects on another subsystem, however, had not been assessed because the hardware was not justified on the basis of its improvements to that subsystem. Subsequent analysis showed significant effects in the second system, and we needed to extend the analysis to reflect the effects on all subsystems.

MAKE ESTIMATES OF RESOURCE CONSUMPTION EARLY -- EVEN BEFORE INFORMATION IS AVAILABLE TO MAKE ACCURATE ESTIMATES

A Performance Engineer must focus attention on the most important performance issues as early as possible. Making early estimates of resource usage is difficult, and accurate estimation calls for experience both with the application and with making estimates. However, even imprecise estimates are very useful for identifying performance problems that will exist when the most optimistic estimate is used.

The difficulty with making estimates is usually a reflection of the difficulty of coming to grips with the important issues of performance design. If a Performance Engineer wants designers to address performance problems throughout the life of the project, he must undertake the effort to work with designers and decide on estimates for the use of each important system resource and for response times. We have repeatedly found that we need to suggest estimates to developers after discussing the design with them. System design is their skill, performance estimation and projection are ours.

TAKE MEASUREMENTS OF REAL SYSTEM PERFORMANCE AS SOON AS POSSIBLE

As implementation proceeds, measurement will eventually be possible, and should begin to replace estimates. If you've been working for months on estimates and models, you'll develop a style of working with system implementors. Switching from that style (involving estimates and models) to a style appropriate for measurement tends to require a force of will. In one case, the first measurement by a Performance Engineer came rather late, and indicated that the system's performance was drastically different from the models. Recovering performance at that late date was difficult and expensive. Earlier measurement was needed.

We have observed that some subsystems managers tend to resist early measurement. They believe that performance will improve after additional work is done and that they may be embarrassed by early results. Acquiescing to their requests for delay will invite problems to grow. Even if no problems were present initially (that must have happened sometime), performance will tend to degrade when no means is used to detect problems so that developers can correct those problems. Functional debugging is a fact of life, and so is performance debugging.

On the other hand, if the Performance Engineer takes the opportunity presented by early data to be arrogant and exclaim about the incompetence of designers, he will find that obtaining aid for making measurements will be increasingly difficult. The trick is to work with developers to aid them with the data before revealing adverse results to their superiors. At this stage of the project, the Performance Engineer's responsibility is to aid designers to improve their work. If radical changes appear to be needed, the situation should be discussed with developers before it is revealed to others.

We have found that early performance results are frequently frightening, and that upper managers can easily be panicked by the results. Although the Performance Engineer should not take the position of defending inadequate results, he should concentrate on ensuring that performance results are interpreted in the context of the development environment and the probable improvement resulting from further efforts.

CONCLUSION

Performance Engineering can pay off handsomely; it can ease the attainment of acceptable performance; it can significantly improve the probability of success in large software developments. However, it requires that problems in the real environment be recognized and addressed.

Practical measures must be taken to make Performance Engineering successful. Our experience is that working closely with developers is critical, that simple models must be employed, and that measurements should replace estimates as early as possible.

As the field continues to develop, we hope that others will document the lessons that they have learned. In addition to the academic literature, we need information from other Engineers that are facing the difficult problems in the field.

REFERENCES

1. Smith, C. U., "Increasing Information Systems Productivity By Software Performance Engineering", Transactions of CMG, No. 49 (September 1985), pp 16 - 22.

2. Levy, A. I., and V. C. Soder, "Performance Analysis During Application Design: An IMS Case Study", Conference Proceedings of CMG '86, pp 167 - 172.

3. Bell, T. E., "Performance Engineering: Doing It 'Later' On Large Projects", Transactions of CMG, No. 55 (Winter 1987), pp 75 - 82.

4. Smith, C. U., "Performance Engineering: A Bibliography", Transactions of CMG, No. 55 (Winter 1987), pp 115 - 122.

5. Lindsay, D.S. and T. E. Bell, "Directed Benchmarks For CPU Architecture Evaluation", Conference Proceedings of CMG '86, pp 379 - 385.

SERVICE LEVEL AGREEMENTS:
"Good Fences Make Good Neighbors"

George W. (Bill) Miller
American Airlines

ABSTRACT

A Service Level Agreement is a written contract between the user and the provider of a service. It identifies the parties to the agreement. It states the user's expectations regarding the volume, availability, and timeliness of the service, and the provider's obligations in supplying the service. It specifies the price of the service if appropriate, and the penalties to be assessed if volumes and/or services are outside of agreed limits. A centralized SLA control function provides a repository of SLA information and coordinates the administration of Service Level Agreements. This paper will define and discuss the key elements which make up a Service Level Agreement, as well as those related activities which must be present and functioning to allow the effective management of the installation's workload to include: Monitoring Procedures, Chargeback, Capacity Planning Procedures, and Organizational Structure. Also included is a proposed general format for Service Level Agreements and a checklist to assist in the inclusion of the proper elements for any given situation.

INTRODUCTION

The management of a large computer can be likened to the challenge of restoring order after a prolonged civil war. Everyone has chosen sides, old prejudices are cherished, and general suspicion is the order of the day. How does one replace this pervasive animosity with a sense of common purpose? The answer has something to do with mutual expectations.

Most of our working relationships can be seen in terms of a service to be provided, and the parties involved as the providers and users of that service. What a properly constructed Service Level Agreement (SLA) really does is to create a mutual understanding of the requirements and capabilities of both parties, and allows them to arrive at a realistic set of common expectations regarding the service to be provided.

In classic DP terms, computer service is usually seen as the processing of batch jobs or interactive transactions and it is this type of service that is most frequently the subject of a Service Level Agreement. Actually, in most working relationships, one can identify a service being provided to the user of that service. Certainly Computer Operations provides the service of controlling the daily utilization of the installation's computer resources by those who need the information provided by the various computer applications.

But just as surely, those with problems need the assistance of the help desk or the coverage group, various business elements need the services of the applications development people, and the operations people rely on the hardware planning people to assure the environment adequate to accommodate the installation's workload. Considered in this manner it is soon clear that most relationships can be described in terms of a service between a provider and a user or customer. We often tend to refer to internal "users" and outside "customers" although the two terms are basically interchangeable. The range of relationships for which SLA's should be constructed is a management decision. The point is not that all relationships should be covered by SLA's, but that most relationships could be.

It should be clearly understood that Service Level Agreements are one part of an overall installation management program which includes monitoring and historical analysis, capacity planning, and chargeback. The successful implementation of Service Level Agreements will require total organizational commitment.

Service Level Agreements:

We can define a Service Level Agreement (SLA) as a written agreement between the customer and the provider of a service which identifies the parties to the agreement, the customer's expectations regarding the nature of the service, the provider's obligations in supplying the service, and, when appropriate, the compensation for this exchange.

For purposes of discussion it is useful to structure the SLA into a number of specific elements even though these elements are largely interrelated.

1. Identify the parties to the agreement.
2. Describe the service to be provided.
3. Specify the volume of demand for service over time.
4. Define the timeliness requirements for the service.
5. Discuss the accuracy requirements.
6. Specify the availability of the service required.
7. Define the reliability of the service provided.
8. Identify the limitations to the service that are acceptable.
9. Quantify the compensation for providing the service.
10. Describe the measurement procedures to be used.
11. Set a date for renegotiation of the agreement

1. PARTIES TO THE AGREEMENT

Depending on the scope of the agreement, the parties to the SLA may be groups within the same department, between major departments, or with customers outside the organization. In any case, agreements are best set out between individuals, either by name or by title, along with their physical location. After all, somebody is going to have to sign the thing.

2. SERVICE TO BE PROVIDED

For purposes of discussion it seems useful to define some terms.

Computer Resource Units (CRUs) are the basic units representing computer hardware activity. Examples would be CPU time, various I/O counts, tape mounts,

DASD storage, etc. Service Units (SUs) are weighted composites of several CRUs and are used to describe the overall hardware resource associated with some activity.

Computer Workload Units (CWUs) are the basic workload elements within the computing environment. Examples are batch jobs and interactive transactions. The resource characteristics of CWUs are described in terms of CRUs, i.e., we may describe the activity of a batch job in terms of the volume of CPU time and I/O counts.

Business Work Units (BWUs) are the business related activities that are the purpose of all this computer activity. Examples are bank deposits, claims received, units manufactured, or passengers boarded. Until we look at the work done in terms of BWUs, we have difficulty in distinguishing the computer installation of a bank from that of an insurance company or a manufacturing company or an airline.

The user of DP service may wish to describe activity in terms of Business Work Units (BWUs) of passengers boarded, etc. The provider of the service sees the activity generated by the conduct of the user's business in terms of a number of the Computer Work Units (CWUs) of jobs, transactions, or messages required to accomplish a passenger boarding. These computer related work units can in turn be translated into Computer Resource Units (CRUs) of CPU time, peripheral usage, etc. The Computer Work Units provide the necessary bridge between work as seen by the customer and demand as seen by the provider. We will look at these relationships again when we discuss pricing.

3. VOLUME OF SERVICE

It is not reasonable to expect that the provider of a service be required to agree to a specific quality of service without knowing the quantity of service which is to be demanded.

Both the expected volume and the time distribution of that volume must be considered and specified. Requirements should specify the average and peak rates at which this work will enter the system and the times during which this demand is expected to occur. The user should be provided with the incentive of better service at a reduced cost by avoiding peak usage periods.

It should be clear that the volume of demand is as much a part of the binding SLA as any other element. The quality or timeliness of service is agreed to be provided for a certain volume of demand at a certain time of day. If the actual volume of demand exceeds these estimates, then it is very appropriate to describe a reduced requirement for quality.

In addition, the customer should be required to define the expectation for the volume of service to be required over some significant period of time into the future. It is not unreasonable to ask for an estimate of volume of demand by month for the next 24 months. The provider cannot be expected to provide a high level of quality service over an indefinite period of time without knowing what is coming next. One of the most tangible benefits to be derived from the development of comprehensive SLAs is the information on expected workload growth that results.

This is particularly important during a time of the rapid incremental growth which would result from aggressive marketing of data processing services. In this situation an accurate estimate of demand from the customer is essential. The accuracy of the estimate may be significantly improved if the customer is allowed to describe demand in terms of Business Work Units. The added constraints on service requirements imposed by the service level agreement provide additional means for providing quality service in a rapidly growing environment.

Reference the fine paper by John Propst. (see [1]).

4. TIMELINESS OF SERVICE

The timeliness metric defined in the SLA will depend on the nature of the work. Discussion with the customer and analysis by the provider will have determined the number and kinds of Computer Workload Units which will be required to support the customer's Business Work Units. These computer workload units may be batch jobs or on-line transactions or input messages. Timeliness will probably be defined in terms of the Computer Workload Units, that is in terms of batch turnaround time or transaction or message response time.

Batch turnaround time may be stated as job start to job stop. For TSO submitted jobs, particularly applications development and user submitted production, distributions of the form "90% within 2 hours" are appropriate. Specific times will vary by class. Additionally for production jobs some times outside the host may be needed, e.g., input available to job submit, or print complete to delivered to the user. Scheduled production jobs may define service in terms of whether the scheduled completion time has been met, e.g., if input is available by 8:00 p.m. then output will be delivered to the user by 8:00 a.m., etc.

Transaction or input message response time as perceived by the user is generally the time from when the enter key is pressed until the keyboard unlocks and a response is received. Measurement of times outside the host processor present measurement challenges although a variety of tools and techniques exist for measuring or estimating these times.

Transactions with consistent characteristics are adequately described by a "90% within 2 seconds" distribution. Some transactions while consistent in their characteristics may be highly data-dependent. In such cases it may be necessary to link response times to the amount of data being processed, and define response time limits which relate to ranges of volume.

A number of examples of timeliness for non DP specific relationships can be found in the SLA checklist found in appendix A.

5. ACCURACY OF SERVICE

A discussion of accuracy generally relates to some minimally acceptable level of defects in the service provided. For batch jobs, for example, this may involve errors in the input data or in some frequency of reruns. In general, computers are impressively accurate, people are not. A number of examples are provided in the SLA checklist.

6. AVAILABILITY OF SERVICE

As an element of a SLA, availability is that time window during which service is to be provided to the customer. A frequently complicating factor is that a system may be up and accessible, but operating in a degraded manner.

The customer's notion of availability is that if work is going in and coming out at the usual rate, the system is available; otherwise, it is not. In fact, availability is the product of a series of system

components, any one of which can affect end user availability. These components include:

- local terminal equipment
- network equipment
- front end processor
- network control software
- mainframe processor
- control units
- I/O devices
- operating system software
- operational support software
- applications programs
- individual databases.

For each of these there is the possibility that a failure will result in loss of availability to at least some users. Obviously the probability and impact of component failure will vary with component. A Component Failure Impact Analysis (CFIA) is a process of identifying each of the components having the potential for affecting end user availability. The probability of failure for each component should be assessed, as well as the expected recovery time. Part of this process would be to verify that appropriate backup and recovery procedures exist for each.

In the event that significant outages do occur, consideration must be given to the most appropriate use of the available service. The SLA should establish the Disaster Recovery Priority for a particular application or user. Depending on the resource available and the needs of the user, this priority might be part of a pricing discussion.

This discussion owes much to an excellent paper by Tony Clifford-Winters. (See [2]).

7. RELIABILITY OF SERVICE

There is often some confusion as to just what is meant by the terms accuracy, availability, and reliability. We have discussed accuracy as a measure of defects and availability as a time window during which service is planned or scheduled to be provided. Reliability has to do with the comparison of actual availability to planned availability. There needs to be an acceptable definition of the conditions that accurately indicate whether or not the system is or is not fully available. Then reliability might most simply be stated as the ratio of the time that the system was actually available to the time during which the system was planned to be available. We would probably state this as a percent, and set a standard for an acceptable degree of reliability.

8. LIMITATIONS OF SERVICE

Customers do not operate in an environment of unlimited resources, nor do they operate alone. There are limits to the level of service which can be provided by the host environment even under optimal conditions. If the user requires a higher level of service than is currently available, then some interim agreements will be required while the environment is evolved to one with greater capabilities.

To the extent that applications share resources, they have the potential for interfering with each other. While some of these problems may be resolved by appropriate costing, there may be constraints imposed on the use of certain resources to enable various applications to coexist peacefully.

9. COMPENSATION FOR SERVICE

If Service Level Agreements are expected to be taken seriously, they should have some teeth. Cost allocation activities that go to great lengths to determine a customer's share of the cost of the installation and then report that cost for the customer's information are relatively worthless. High level summaries reported to corporate management can get some temporary attention, but this is a crude way to manage costs.

The way to keep us all honest requires a real live chargeback system. by chargeback is meant a procedure for not just attributing but billing the cost of providing computer service to the user of that service.

Service may be internal to the corporation or may be sold to an outside customer. It can be said that these present different goals: the goal of internal usage being to minimize costs by limiting demand, the goal of selling service to outside users being to maximize profits by encouraging demand. However, if it is accepted that the intelligent use of data processing is an activity to be encouraged, then both cases involve providing high quality service at a minimal (competitive) cost. The factors to be determined are units and prices.

The development of effective pricing structures will be determined by the need to recoup costs and the desire to establish a profit margin. When the corporation is its own sole user of DP service, the level of profit built into charges is essentially an accounting consideration. When the data center is selling service to an outside customer, the goal is to maximize profit while providing an attractive quality at a competitive price.

While the specific price structure can only be determined by careful financial analysis, it is useful to review some general principles.

Breakeven pricing is based on the notion that the total cost of the data center resource should be borne by the users of that resource. Each user pays a proportionate share of the total. The result is that charges will vary from period to period. A user's cost for doing the same work will go up when total utilization goes down and would decline when utilization is high. This discourages the user from using the resources to best advantage and makes planning very difficult.

Standard costing plus a profit margin assigns a fixed cost to each Computer Resource Unit (CRU) which in turn will fix the cost of a standard Computer Work Unit (CWU) which in turn will fix the cost of a standard Business Work Unit (BWU). The point here is that we are agreeing to use averages which will allow us to fix rates. We estimate the average cost for a unit of CPU time, or for I/O activity, or for some unit of DASD storage, and use these rates regardless of the specific level of utilization for a given period. We can go through a similar process of determining the average number of CRUs required to run a particular class of job or transaction, and arrive at a standard cost for a CWU. Likewise, we can determine that the accomplishment of a particular business activity typically requires a certain number of batch jobs or transactions or whatever, and arrive at a standard cost of servicing a BWU. Since the cost is based on a fixed average, the actual charge for service at any given time is an estimate. The overall results should be reasonable, however, and the advantage to the user of having consistent understandable charges is very significant. Standard costs can be revised when the SLA is renegotiated.

While this discussion is in terms of hardware resources, there are obviously other costs. The related people and environmental costs are often associated to hardware resources in terms of cost pools. The specifics of such things are proper topics of a comprehensive discussion of chargeback, but are somewhat beyond the scope of this discussion. For example, suppose that a travel agency can estimate the volume of passenger bookings (BWU) that it expects to occur in a given period of time. The agency contracts with some high class airline to provide them with the required service. Say, just for sake of an example, with American Airlines. After some analysis of the kind of business this agency does, it is determined that they will require an average of 20 transactions to accomplish a booking. The appropriate cost to process this activity is developed based on standard costing plus some margin for growth and profit. Charges are billed to the user based on the number of bookings accomplished. The customers cost is directly tied to the volume of business. The provider's cost will be more or less than planned largely according to whether the average number of transactions required to do the booking is more or less than 20. As these relationships change over time, the price structure will need to be renegotiated. Price incentives can be used to encourage greater efficiency on the part of the customer. In this way the customer has some influence over the cost of his service, and may consider the option of choosing to trade off DP expense for employee skill level.

Standard costing has a number of advantages:

- the customer can project volume in Business Work Units
- the customer is billed based on the amount of Business Work Units generated
- the cost of a Business Work Unit is fixed for the life of a Service Level Agreement
- the customer can reduce costs by requiring fewer Computer Work Units
- the provider can increase profits by operating more efficiently.

The primary difference in internal users and outside customers is in the method of payment. Internally, all this can get real in a hurry if the cost for the service is part of the user's budget.

In a non price-specific arrangement, compensation can still take place in terms of nonmonetary incentives and/or penalties. Compensation may relate to wage and bonus incentives or to the basic MBO process.

OPTIONS: The relating of appropriate charges to both the volume and quality of that service gives the user an incentive to utilize service as effectively as possible, and gives the provider an incentive to improve quality. For this reason it is necessary to tie service level costing to both volume and service.

Rates should be highest for demands made during prime shift, perhaps higher still during the peak mid-morning and mid-afternoon hours.

Disaster recovery priority or priority during periods of limited resource availability should require some additional financial compensation.

Batch turnaround times or interactive response times which are near the ultimate limits of the host environment should warrant premium rates, while more modest levels of service will cost less.

The implication here is that a user who demands premium service will expect to pay premium rates, and the user whose demands are more modest can expect to save money. The corollary is that if the user is provided with a level of service that is less then that for which he contracted, he is entitled to pay a lower rate.

Some very good discussions of costing are referenced. (See [3] [4]).

10. MEASUREMENT OF SERVICE

Comprehensive monitoring procedures are central to the establishment and maintenance of SLAs and to the effective management of the data center as a whole.

Proper monitoring procedures begin with the means for routinely obtaining installation performance information. Sources of such information include SMF files, RMF data, IMS logs, hardware monitors, network monitors, and manual logs. This collective information forms the basis for everything known about what has happened in the data center.

The large volume of this very detailed information requires that it be summarized for historical purposes. this summarized history is one of the single most valuable resources possessed by the corporation. Analysis of this information makes possible:

- the intelligent establishment of achievable Service Level Agreements
- the accurate assessment of charges to the user
- the performance tuning information required to insure that the environment makes the maximum resource available
- the historical trending which allows reliable forecasting
- the detailed modeling information required to support hardware planning.

The importance of this historical information to the corporation implies that the primary tapping of the raw sources and their initial summarization should be a consolidated activity rather than a separate effort for each area of interest. There should be a Performance Data Base (PDB) to serve as the source for this information by those requiring it.

Consolidation of this activity eliminates considerable duplicate effort both in manpower and in the significant amount of computer resource required to process these very large accounting files. An even greater benefit is the sharing of a single, comprehensive, and consistent source of historical information. Common access to an expanded fund of knowledge will enable more intelligent decision making.

The consolidation of these various activities requires a high degree of coordination to insure that all required information is present in the master performance data base, and that maximum advantage is taken of new tools and technologies in the field of performance monitoring.

12. RENEGOTIATION OF SERVICE

Under normal circumstances, service level agreements should be renegotiated at least annually as a part of the budget cycle. As a practical matter, they may need to be reviewed more often.

Some deviation from standard can be expected as a part of the natural dynamics of a large system, but SLAs which are significantly out of bounds for several months in a row should be renegotiated immediately.

Significant changes in the host environment or in the needs of the user would require immediate renegotiation.

Organizational Responsibility:

Organizationally, a centralized SLA control function should be established to provide a repository of SLA information. This central control would have the responsibility for chairing the SLA negotiating team. Members would include representatives of both the user and the provider of the service, and others whose involvement requires their participation. The centralized control function should also provide the coordination required to incorporate the various SLAs into the data processing limitations of the installation as a whole, and to allow their coexistence.

Responsibility should be explicity assigned for the following critical activities:

- to coordinate and assure the routine collection and archival of essential performance information, i.e., "the keeper of the data"
- the determination and periodic review of standard costs
- to assess the service level capabilities of the host environment as they form limitations on individual Service Level Agreements
- coordinate the creation of new Service Level Agreements and their subsequent review and renegotiation
- maintain a centralized repository of Service Level Agreements
- coordinate the creation of a Component Failure Impact Analysis (CFIA) to support availability service levels
- coordinate the creation of intra-organizational SLAs to support the CFIA activity
- generally facilitate the SLA process in a controlled, consistent, and unbiased manner

Conclusion:

The establishment of internal Service Level Agreements as a routine way of doing business will cause a significant impact to all areas of the corporation. Requirements, expectations and procedures will have to be discussed and formally defined. The eventual result will be to permit a greater degree of control over the cost and growth of services.

As a practical matter, the parties to the exchange will eventually come to a set of mutual expectations with an SLA. What the formal Service Level Agreement will do is to allow this to happen initially as a result of negotiation, rather then wait until differences in expectations have deepened into bitter disputes that force decision making at inappropriately high levels in the organization. Pay now or pay later.

REFERENCES

[1] Propst, John W., "Developing Effective Service Level Agreements," Conference Proceedings of Computer Measurement Group (December 1985), pp. 704-709.

[2] Clifford-Winters, Tony, "Service Level Agreements: To Negotiate or Impose," Conference Proceedings of UK Computer Measurement Group (May 1986), pp. 37-59.

[3] Wenk, Dennis, "Management Control of Management Information Systems," Conference Proceedings of Computer Measurement Group (December 1985), pp. 486-489.

[4] Howard, Phillip C., "Standard Costing in Data Processing," EDP Performance Review (June 1981), pp. 1-6.

APPENDIX A.

SERVICE LEVEL AGREEMENTS - Issues and Options

A Service Level Agreement (SLA) is a written agreement between the customer and the provider of a service. It identifies the parties, the customer's expectations regarding the nature of the service, the provider's obligations in supplying the service, and, when appropriate, the compensation for this exchange.

There are five major issues to be resolved in the development of Service Level Agreements:

1. Services to be covered by the agreement
   - classic central site/DP customer services
   - applications development
   - technical assistance
   - administrative support
   - training

2. Parties to the agreement
   - outside customer
   - internal user
   - within Data Processing
     DP will require hierarchy of SLAs for support
     within department groups
     across department boundaries

3. Development of the agreement
   - negotiating team
     customer(s)/provider(s)
     SLA coordinator (responsibility)
   - complete and detailed service definitions
     nature/volume/timeliness/quality
     standards and means of measuring
     incentives/penalties

4. Enforcement of the agreement
   - pricing structure (if appropriate)
     cost unit should relate to service provided
     price relates to both volume and quality of service
   - effective chargeback system required
   - non-price incentives/penalties
     MBO review item
   - capabilities for measurement and reporting must exist

5. Review of the agreement
   - schedule of periodic review and renegotiation
   - adjustments to volume/service/incentives

A proposed general format for Service Level Agreements and a detailed checklist follows:

SERVICE LEVEL AGREEMENT	Reference Number

This document constitutes a formal contract between
_____ (Provider)
and
_____ (Customer)
The specific support objectives and limits to the services provided are described within this document and its attachments.

Implementation Date	Review Date
Provider's Location	Provider's Coordinator
Customer's Location	Customer's Coordinator
Description of Service	
Volume	
Timeliness	
Accuracy	
Availability	
Reliability	
Limitations	
Compensation	
Measurement	

Provider Approval	Date	Customer Approval	Date

SERVICE LEVEL AGREEMENT CHECKLIST

This checklist is intended to assist in the development of a Service Level Agreement. Since Service Level Agreements can be used to describe a variety of services and parties, the particular elements appropriate to a specific SLA will depend on the circumstances.

Implementation date
- Effective date for this agreement

Review date
- Date or period for planned review of this agreement

Define parties to agreement
- Provider of service
- Provider location(s)
- Provider coordinator
- customer for service
- customer location(s)
- customer coordinator

Describe service to be provided
- Purpose of service
- Location of service
- Type of service
  - Commercial DP services
    - Batch jobs
    - TSO development (SPF)
    - IMS
    - DB2
    - APL
  - Real time DP services
    - ACP/TPF
    - VM/CMS
    - Other
  - Communications services
    - Terminals
    - Concentrators
    - Front-End
    - Lines
    - Modems
    - Network control
  - Applications development services
    - Functional review
    - Design
    - Testing
    - Implementation
  - Technical assistance
    - Computer operations
    - Software support
    - Control program support
    - Management reporting
    - Workload forecasting
    - Hardware planning
    - Strategic planning
    - Contract administration
    - Hardware installation
  - Problem resolution
    - Coverage
    - Help desk
  - Maintenance
    - Hardware
    - Software
    - DP internal
    - Outside vendor
  - Training
    - For DP
    - For customer
    - By DP Training
    - By outside vendor
  - Administrative support
    - Personnel administration
    - Medical support
    - Social services
    - Insurance
    - Employment services
    - Financial services
    - Procurement
  - Environmental support
    - Facilities
    - Security

Volume Expectations
- Batch jobs
  - Weekly totals - min, max, and average
  - Daily peaks - min, max, and average
  - Distribution over prime, non-prime and weekend
  - Projected 2-year growth
- Interactive transactions/messages
  - Daily totals - min, max, and average
  - Hourly peaks - min, max and average
  - Hourly distribution over prime, non-prime, and weekends
  - Projected 2-year growth
  - Terminal Count
  - Terminal Growth
- Ceiling on maximum or peak volumes
- Communications network volumes
  - Daily totals - min, max, and average
  - Hourly peaks - min, max, and average
  - Hourly distribution over prime, non-prime, and weekends
  - Projected 2-year growth
  - Terminal Count
  - Terminal Growth
- Applications development
  - Functional specifications
  - Manning levels
  - Testing requirements
  - Documentation
  - Implementation schedules
- Technical assistance
  - Specify services to be provided
  - Reports
  - Forecasts
  - Planning documents
- Problem resolution
  - Support services to be provided
  - Volume of demand for service
  - Daily totals - min, max, and average
  - Hourly peaks - min, max, and average
  - Hourly distribution over prime, non-prime, and weekends
  - Projected 2-year growth
- Maintenance services
  - Maintenance services to be provided
  - Expected volume/frequency
  - Scheduled/unscheduled maintenance/modifications/releases
- Training
  - Type of training to be provided
  - Training schedule
  - Class size/frequency
  - Location

Administrative support
- Type of service to be provided
- Expected volume/frequency

Environmental Support
- Type of service to be provided
- Scope/volume/frequency

Timeliness Expectations

Environmental capability provides limits

Batch jobs
- Turnaround times
  - Elapsed job start to job stop
  - Input received to output delivered windows
  - Average times vs. percentile distribution
  - Volume related ranges
- Conformance to schedule

Interactive transactions/messages
- Response times
  - Host processor times
  - Customer terminal times
  - Average times vs. percentile ranges
  - Volume related ranges

Communications services
- Network delay times

Applications development
- Conformance to scheduled implementation dates

Technical assistance
- Conformance to scheduled implementation dates
- Established frequency/publication dates
- Response to crises demands

Problem resolution assistance
- Wait time for service initiation
- Time for problem resolution

Maintenance
- Conformance to maintenance schedule
- Response to crises demands

Training
- Adherence to schedule
- Training duration

Administrative support
- Wait time for service initiation
- Elapsed time for service completion

Environmental support
- Adherence to schedule
- Response to crises demands

Accuracy

Accuracy relates to defects in service performed

Generally input-related for batch processing
- Determination of responsibility
- Rerun policies

Applications development
- Defects per unit of time
- Mean time between defects at implementation
- Elapsed time between implementation and stabilization

Technical assistance
- Defects per unit of time
- Accuracy of information
- Actual vs. forecast variance
- Repeats/revisions required

Problem resolution
- Accuracy of information
- Retries required

Training
- Test scores vs. standard
- Retraining required

Administrative support
- Defects per unit of time
- Repeats/revisions required

Environmental support
- Defects/violations per unit of time
- Corrections required

Availability

Commercial/Real time primary DP services
- Time period during which service is to be provided

Application development
- Project length

Problem resolution/maintenance/ environmental support
- Time period during which service is to be provided

Technical assistance/administrative support
- Normal office hours

Training
- Course schedule

Reliability

Reliability = actual availability as a percent of scheduled availability

Conditions and exceptions
- Perception of customer and provider may vary
- Periods of limited capability
- Frequency of interruptions

Limitations

Interdependencies

Impact on other areas
- Resource constraints
- Conflict with shared resources

Priority
- With respect to provider's other work
- With respect to customer's other work

Backup requirements

Security requirements

Disaster recovery policy and procedures

Cost

Definition of pricing unit(s)
- Charge for service provided, not resource consumed
- Cost of service fixed by provider
  - Computer resources
  - Man-hours
  - Materials
  - Related services
  - Profit margin

Relation of price to service
- Premium price for premium (faster) service
- Price adjustment for substandard service
- Price discounts for off-peak times

Relation of price to volume
- Decreasing scale to encourage volume
- Price penalties for excessive peak or peak-time volumes

Nonmonetary incentives/penalties
- Wage and bonus incentives
- May tie to MBO

Measurement

Responsibility
- Provider
- Customer
- Shared

Appropriate metrics

Source of information

Measurement tools

Analytical tools

Definitions

Assumptions

Approximations

Reporting vehicle

Reporting audience

Reporting frequency

CAPACITY PLANNING IN THE AGE OF COMPETITIVE ADVANTAGE COMPUTING

Dr. Robert P. Goldberg
Dr. Leonard D. Lipner
BGS Systems, Inc.
128 Technology Center
Waltham, MA, USA 02254-9111

ABSTRACT

As information becomes the driving force behind company operations, and as it affects the relation between the company and the outside world, the Information Systems (I.S.) function has come to be perceived as a competitive weapon. I.S. today is often viewed as a critical partner to strategic business units, and even to company survival.

In case after case, the new computing environment has made extraodinary demands on the Capacity Planner. While having to cope with all the traditional CP issues, he is faced at the same time with a whole new set of problems. Many of these call for new skills, and sometimes even an expanded definition of Capacity Planning.

This paper describes the new "competitive advantage" computing environment, and its implications for the Capacity Planner. Using examples drawn from experience, we illustrate the new kinds of technical and organizational problems he faces, and the new skills he needs in order to solve them.

The Age of Competitive Advantage Computing

In a free market, successful businesses have always had to pay attention to the realities of competition. Like it or not, those who evaluate and shape business practices have had to keep the competitive environment in mind, at all times.

New business practices are usually either attempts to gain a competitive edge, or responses to such thrusts. In fact, one can often read the history of an industry primarily in terms of the competitive thrusts by its innovators and the responses they generate. While specifics vary from one industry to another, competitive thrusts and responses share a relatively small set of objectives. For instance:

1. To lower the costs of production, or at least to keep them from rising, by:
   a. Reducing or controlling the cost of labor, raw materials, and equipment, or
   b. Improving the efficiency of company operations (in design, manufacturing, warehousing, marketing, sales, accounting, etc.).
2. To increase revenues, or at least to keep them from declining, by:
   a. Increasing market share, for example by differentiating one's products in terms of unique added value, by superior products, or by a superior substitute product,
   b. Increasing the size of the market by identifying and selling to new customer groups,
   c. Charging a premium for features the market perceives to have special value,
   d. Developing and bringing to market entirely new products, especially those which leverage one's current strengths,
   e. Reducing the ability of customers to switch to another product or vendor, by creating a situation in which the cost of switching is high,
   f. Reducing the desire of customers to switch, by offering better products and/or services at comparable cost, or
   g. Reducing the ability of other companies to compete, by creating high barriers.

In its early days, computing had little influence in any of these competitive areas. More often than not, the original purpose of computing was to automate accounting, and, later, other administrative (i.e, record-keeping) functions: orders, manufacturing, personnel, etc. All of these uses were aimed at reducing costs rather than increasing revenues. In most businesses, however, administration as such represented a relatively small percentage of a company's expenses, and still does. So, while computers and automation produced sufficient savings to justify their acquisition (even when computers were expensive), these savings had little impact on the company's overall marketplace success or failure.

Online systems, like those for airline reservations or customer inquiry systems, were the first computer applications to influence company strategy and success in a major way, because they were developed not only to cut administrative costs, but to offer differentiated customer service, and thereby to increase revenue substantially. As they grew in size and function, it gradually became apparent that they could (if planned correctly and delivered on time) produce virtually all of the revenue-increasing competitive advantages listed above. Of course, it takes time to uncover and develop the applications with the most competitive clout; it doesn't happen in a day. But with the first development of online systems, the age of competitive advantage computing had begun.

This evolution away from the older view of I.S., as "back office administrative support," occurred gradually. However, it has accelerated in the recent past for three main reasons:

- the dispersion of computing power which put access to data into the hands of end users,
- the relative ease with which users can try out new ways of processing and viewing data to support their business activities, and
- most important of all, the competitive environment: the perception that the competition may well be using I.S., or planning to use it, to gain some advantage in the marketplace.

Today, end user computing is widespread, and its value (real and potential) is almost universally acknowledged. Up-to-date, centrally maintained data is often readily

available directly to business units, along with extremely rich English-like facilities for retrieving, combining, processing, distributing and displaying it. While most large companies still retain large central DP staffs for development and operations, expertise and computing power are now well dispersed in the user community. Since in this case familiarity breeds comfort, the days of "computerphobia" are just about over, at least among middle management. So, it's only natural to expect a great deal of experimental thought, and actual experimentation, regarding new uses of existing information in business. Indeed, that's what is happening.

Examples of Competitive Advantage Computing

It's hardly possible to read the DP trade press these days without seeing at least one item about a company which benefitted by discovering a new way to use its information resources. Often the benefit is small, but sometimes it has a major impact on the company's competitive posture. In a few extreme cases, the competitive thrust is so effective that it actually changes the structure of the industry in which the company operates. More often than not the performance of such applications is critical to their success.

It is now established that for almost every aspect of a company's operations, and every aspect of its relations with the outside world (e.g., with suppliers, customers, rivals, regulators, and the public), some company somewhere has found a way to use information to improve that aspect, and as a result, improve its competitive position. Competitive advantage systems can help a company in a variety of ways:

- They can lower those costs which are major contributors to overall product cost, helping the company become (or remain) the lowest priced vendor.
- They can improve the company's ability to respond to customer orders more quickly and reliably than the competition, often a major differentiating factor in commodity businesses.
- They can materially improve the quality of customer service, before, during and after the sale. This improvement can constitute a critical difference between companies, especially in those industries where such service is a major evaluation criterion for buyers.
- They can lock in customers, drastically reducing their desire to switch to another vendor, in part because the price of switching is high, and for other reasons as well.
- They can add significant value to existing products, for example where the customer has a need for information related to his use of a product, and the company is in a position to supply part or even all of that information, along with the original product. This is especially important in industries where there are few differences among competing suppliers.
- They can establish dominance in a niche marketplace, thereby erecting extremely high barriers for competitors.
- Information or information systems can themselves be marketed and sold, constituting new product opportunities for companies which have developed them for other purposes anyway.

A few specific examples will illustrate the diversity of competitive advantage systems. At the same time, these examples are chosen to show the (often unreasonable) pressures such systems can have on the Capacity Planner, who must somehow ensure that they'll have the power they need, when they need it, to perform as expected.

* Example 1: Airline Merger

The deregulation of the airline industry was the starter's gun in an acquisitions race among carriers seeking to benefit dramatically from economies of scale. Merging two airlines' information systems (and more specifically their reservations systems) is almost always an integral part of making such acquisitions work. While not normally a hard problem, the merger of two large and almost equal airlines led to extraodinary Capacity Planning issues.

Both airlines' original reservations systems were based on IBM's TPF, a high-performance transaction processing SCP which currently lacks multiprocessor support. So, the first question for the Capacity Planner was simply whether the combined workloads, PLUS forecasted volume growth, PLUS anticipated enhancements made necessary by other aspects of the competitive environment, would all run on any available uniprocessor. The answer, apparently, was NO.

Senior management made it clear that merging the reservations systems was a MUST for the merger to succeed. As a result, the next series of capacity planing questions involved evaluating the options available to make the combined TPF systems work:

- Was there a way to "split" the TPF system to permit two copies to run on two CPU's and to share the transaction load? One copy would own the databases and provide service functions to the other copy as, for example, in CICS MRO or ISC. Given the anticipated transaction volumes, and enhancements to the TPF system, would the two copies be able to handle the work?
- Was it feasible to consider modifying TPF (drastically) to provide multiprocessor support? What were the implications of entering what was essentially the SCP software business? Would it be possible to find/keep the people necessary to maintain it? What were the true costs, (initial and ongoing) of this approach, and did the benefits outweigh them?
- If the new airline went into the SCP business, were there new technologies which could materially reduce development costs (CASE, application generators, 4GL's, relational databases, etc.)? If so, what were THEIR capacity implications?

An important aspect of this problem, which is typical for competitive advantage systems, is the way the capacity planner is necessarily drawn in to I.S. planning discussions far earlier than he may have been in the past. Several factors account for this:

- The critical dependence of company success on the information system in question.
- The unique nature of the information system.
- The magnitude of the information system.
- The high visibility and importance of the information system's performance.

We'll see many of these same characteristics in other examples.

* Example 2: Patchwork Quilt Financial System

Nowhere are competitive advantage systems more in evidence than in the financial services industry. Many deals today are aimed at pooling the specialties of several companies in order to bring to market a multi-function financial service, offering the customer the ability to manage a diverse set of (unlike) financial transactions in a single account.

For example, one group of companies combined their computing expertise and information resources to offer a single account for managing stock, bond, money market and mutual fund transactions, as well as insurance and loan processing. Furthermore, to offer unique value, they provided not merely monthly statements, but online access to account information from PC's in the office or at home. In the mad scramble to establish an edge for their offering, they took the next step as well by combining these features with online access to public financial and other databases, advisory services, etc.

The lure of being first and best in a highly competitive market obscured the difficulty of creating the information systems needed to support such an offering. The mistake was to envision that the in-place systems supporting each of those INDIVIDUAL functions could be "glued together" relatively easily, with consolidated statement processing handled as a kind of "shell" around them.

The architecture of their system was badly flawed. The problem was that they built in, almost by design, critical bottlenecks in the batch component. To state it simply, each component system was scheduled to run "as usual", the one difference being that results of each run were saved in order to feed a critical new "consolidation of results" component. The individual batch components could to some extent be multiprogrammed, though at a price of additional contention and therefore stretchout. As functions were added to each component over time, this contention only got worse.

However, by far the most serious bottleneck was the consolidation run, since (1) it had to wait for all the individual component systems to complete, and (2) no printing of final reports could begin until consolidation processing was itself complete. (Typically, the daily consolidation run accessed ALL accounts in which ANY activity took place.) This one characteristic caused the overnight window to be missed repeatedly, leading to availability problems, and initially, poor customer acceptance of the offering. In the final analysis, it was the impetus for a 6 month delay in bringing an offering with adequate performance to market, due to (a) a requirement for substantial application redesign, and (b) a significant, unexpected capacity upgrade. During that period, competitive offerings appeared, severely blunting the first group's advantage.

In fact, there may have been no other way to bring the offering to market more quickly. Doing the job right, i.e., building consolidated processing functions directly into the individual component systems, may simply have taken longer than the window of opportunity stayed open.

Sometimes, taking too long exposes the risk that a competitor's offering will dominate the market. Their system may be as good, or better, or even worse, as long as it's earlier. If the service it delivers results in even moderate customer loyalty, it will be extremely difficult to compete with later on unless one brings out a substantially better product. They will have effectively upped the ante to an unacceptable level.

* Example 3: Intercompany Networks Using Electronic Data Interchange

Whenever two companies are in a supplier/consumer relationship, and do a substantial amount of routine business with each other, EDI may offer both of them numerous benefits. For example:

- For the supplier:
  - Reduce the cost of sales, by allowing the consumer to enter orders directly rather than through a salesperson.
  - Improve cash flow.
  - Reduce the likelihood that the consumer will switch to another vendor.
  - Many others.
- For the consumer:
  - Speed the order entry process, and make it more reliable.
  - Track order status directly, online, in order to make more informed business decisions about the scheduling of his own work.
  - Lower operating costs, by reducing the need to carry excessive inventories,
  - Many others here too.

There is no question that when set up properly, intercompany networks can deliver significant competitive advantage to all participants.

EDI, however, changes all the ground rules for the capacity planner. Other companies on the EDI network are driven by marketplace forces which are entirely outside his awareness, familiarity and knowledge. Yet those forces can seriously affect the service his users will receive, for two reasons:

- The performance of his computer is, to some extent, dependent on the response time of another company's computers. Since workloads on that other computer are driven by external forces, response times from the EDI-partner's computer may may be erratic, or they may exhibit patterns which are hard to recognize and even harder to predict ("from a distance"). The erratic performance may be caused by forces affecting the EDI workload itself, but in fact it may also be caused by unusual behavior in ANY workload on that other computer.
- Conversely, many of the transactions processed on the capacity planner's own computer originate in other companies. For reasons unknown to him their arrival patterns, and changes in their resource demands may behave in ways he finds inscrutable. So, for the very same reasons mentioned above, a major part of the information he needs may be entirely beyond his grasp.

The problem is even more complex when multiple companies communicate via EDI. Specifically, because of the way overall network performance depends on the performance at each node, a "blip" in one company's business can cause ripples in the computer systems of all of its direct EDI partners, then in all of their EDI partners, and so on.

* Example 4: Marketing In-House Developed I.S. Technology or Data Bases

As Information Systems develop and mature, companies sometimes find that they possess applications or data, originally built to serve their own needs which (they perceive) other companies or individuals may

find valuable. It's only a small step from there to the belief that a profitable business opportunity exists. In many cases, before they know it they've diversified into the I.S. technology business. Almost overnight, the "customers" of I.S. are not only internal users, but real, live PAYING customers.

Examples of such diversification exist in virtually every industry. An insurance company may discover that other insurers are interested in its home-grown system for processing a very unique kind of policy (say, pet insurance). Similarly, a department store chain may develop a system for managing a vast array of deals with trucking and other distribution companies, only to find other chains interested in acquiring the system. Even now, with vertical applications available in many industries, there are still a wealth of niche applications developed by ordinary companies, which are too specialized for software vendors, but still of general interest in the industry. They represent possible businesses.

Another available I.S. resource in many instances is spare computing capacity. Sometimes, companies procure this power as a (sensible, in our opinion) hedge against erratic peaks in important workloads. Sometimes, spare capacity is the result of a terrific upgrade deal. To recoup their expenditures however, companies sometimes try to sell this spare capacity to outside users.

In such an environment, quality of service suddenly takes on a whole new meaning. The capacity planner needs to avoid at least three pitfalls:

- Performance itself may be an integral part of the sales message, and when it is, it has to be delivered no matter what. This can of course lead to distortions in the use of priorities, and other methods of guaranteeing service. Specifically, internal work may suffer dramatically during heavy periods, as the price of delivering agreed-upon service to outside users. It's clear that this question should be addressed up front, and ground rules set which everybody, internal and external users alike, can live with.

- As was the case for EDI arrangements, workloads generated outside the company are always the hardest ones to predict. Users may contract for a certain minimum amount of usage, but they don't know, and won't in general be willing to predict, when that usage will occur. They won't commit to a maximum usage either, preferring instead to view the system as a utility, ready to meet any demand.

- As every software company knows, and as others quickly find out, system maintenance can be a far larger effort even than system development. It can, if one is not careful, place an overwhelming test load on the computer system. Perhaps just as importantly, it can impact one's technical staff, limiting the manpower available for such other capacity-saving activities as system and application tuning.

MORE Unreasonable Pressures on the Capacity Planner

The examples above are only a sampling of the ways competitive advantage computing can change the rules for the capacity planner. To illustrate how diverse these changes are, we list below some additional competitive situations, and their consequences for capacity planning.

- Acquisitions and Divestitures

Unlike the airline example which involved a single large acquisition, some companies follow the strategy of growth by rapid acquisition of many smaller companies. They look especially for those companies which show true synergy with the parent over time, and divest those which don't.

In such a corporation, information systems come and go, and even when management determines that IS functions should be merged, the capacity planner has to be extremely fleet-of-foot, because what is merged today may well be gone tomorrow.

Computing economy of scale is sometimes an integral part of the decision to acquire in the first place, especially in information-intensive businesses. It's clearly important for the capacity planner to be part of the initial disussions about the proposed combined operations, since he has unique insight into the method for determining their true costs.

- World-Wide Systems

As companies extend their markets around the world, it can happen that online systems have to be up for more and more continuous hours each day. This can obviously squeeze any required batch update window during which the online system has to be down. The extreme case, of course, is round-the-clock operation, (i.e., the batch window is zero!) which can be both an applications and capacity planning headache.

- Executive Information Systems

In every case the authors know of, special steps are taken to recognize when the CEO or one of his staff logs on, and to ensure them unlimited highest priority service. The problem is, of course, that both their usage periods and the nature of their work can be erratic. Careless planning can be disastrous to the performance of any other concurrent work on the system. One can always provide spare capacity for just such occasions, but one still has to know how much, and probably has to consider what to do with it at OTHER times.

- Supersystems

These are massive information systems, developed by companies attempting to establish dominance of specialized niche markets, in which information itself is a principal part of the deliverable. Many examples of these systems exist in banking and other financial service areas, such as mutual fund trade processing.

Supersystems have a number of characteristics which make life uncomfortable for everyone connected with them, capacity planners included:

. The window of opportunity is short. Several competitors enter a mad race to get their offering to market first. This almost mandates design and implementation compromises which lead to suboptimal performance.

. It costs a lot to bring a system to market which, due to its breadth of function, erects substantial competitive barriers.

. The risks are therefore enormous. Many situations are "winner take all".

. Both performance and cost-per-transaction are

critical success factors. Among other things, the system has to work at a cost commensurate with revenue.

When they are successful, supersystems also have the characteristic that their growth rates can be overwhelming, as competitors go out of business and large numbers of new customers show up "almost overnight". Getting a handle on reasonable workload forecasts, in time to do something about them, can tax even the nimblest capacity planner.

Skills for The CPM Specialist

It should be apparent from all these examples that the capacity planner who survives and flourishes in the competitive advantage company will have a unique set of skills. Naturally, he will have complete command of all the "classical" planning techniques for:

- Workload measurement, characterization, reporting and forecasting
- Service level management
- Performance modeling and prediction
- Technology forecasting and analysis
- Budget calculations
- Plan preparation and management, plus
- The many details within and alongside these tasks.

To these the planner with a strong sense of career survival should add the following:

* Be quick on your feet

  The best plans are sometimes subverted by an ill-behaved reality that follows its own rules. Let reality come first, and be ready to make major mid-course corrections on a moment's notice. That's what navigating on a stormy sea is all about.

* Develop a working understanding of the processes your company employs for both I.S. and business planning.

  This should be done regardless of how formal or informal those processes are. Determine whether they are done at all, how often, by whom, and most importantly, how integrated or separate they are. In order to develop a sense of where the business is and where it is headed, seek opportunities to discuss user information needs with both business and I.S. planners. Stress the (inherent) interrelationships among business venture planning, planning the information systems which support them, and planning the computer systems which support I.S. Finally, express your willingness to work with them in order to begin to integrate all these planning areas, as they should be.

* Seek a clear statement identifying your company's critical business systems, and their performance requirements.

  There's more to this than meets the eye, since perceptions of what is critical to company survival and what isn't vary a lot among people in the organization.

* Determine if your company's management believes in the use and value of information systems for competitive advantage.

  There are many industries in which information as such is simply NOT a major factor in making a company successful. However, if it is a major factor in your industry, determine the people in your company who think so, and nurture good working relationships with them. It goes without saying that to do this, you have to know a fair amount about the business your company is in: how success is measured, and what makes the marketplace tick. Fortunately, there are usually a great many ways you can find this out.

* Find out your company's strategic vision, and who is charged with communicating and pursuing it.

  Awareness of this vision can completely change the emphasis in your presentation of capacity plans to management. It can show clearly how you're helping the company position itself for success, in terms senior management can understand.

* Determine, and seek concurrence on, a list of "reasonable" I.S contingencies.

  This list should be based first on the most likely competitive thrusts your company will undertake over the planning horizon, and second, on expected thrusts by your competitors and your responses to them. In most cases it will probably be quite difficult to do this, but take your best shot, and use the advice of professionals who spend most of their time watching the marketplace.

* Include analysis of business and I.S. contingencies as a standard part of your capacity plan.

  Based on the list of contingencies described above. Make liberal use of sensitivity analyses, especially to be able to identify "early warning indicators" - a small set of marketplace and I.S factors to watch as early signals of the need for mid-course corrections to the plan.

  Develop a plan for hardware order positions to maximize your ability to respond to sudden requirements for capacity increases (or decreases, in the cases of divestments).

* Develop a tolerance for, and an ability to process, "fuzzy" forecasts. For those forecasts, and for other reasons, appreciate the importance and uses of spare capacity.

Conclusion

This paper describes many (but not all) of the approaches companies are taking today to reap maximum competitive benefit from their information resources. We have seen that in most cases, the capacity planner must be aware of the business environment, and must develop his plan with that environment clearly in mind.

The "classical" techniques for capacity planning can and must still be used. To these the planner must add a whole new set of skills and awarenesses, built upon cooperative relationships with other kinds of planners. These new skills represent maturation and career growth for the capacity planner, and they parallel the integration of I.S. and business planning in the company as a whole.

STRUCTURED MIS PLANNING

Veli-Matti J. Salminen
National Advanced Systems

ABSTRACT

A capacity planning project can easily become an overwhelming effort due to the generally fuzzy definition of what the boundaries of that project should have been. At the same time, businesses are seeking ways to integrate their MIS plans with their business plans, thus further clouding the issue. As a result, MIS capacity planning is changing from a single-person effort to a team effort and from an one-off exercise to an ongoing series of activities across the organizational boundaries.

This paper presents a different look at MIS planning based on solid experience. Identification of required planning components and participants is followed by a framework of managing the planning process across the organizational boundaries.

1.0 INTRODUCTION

The capacity planning process used to be relatively simple in the days of batch-oriented environments and past generations of program development tools, not in terms of required skills in statistical methods, but rather because of predictability of probable changes.

Commercial computing applications, in general, concentrated on automating already well understood manual processes with an objective of reducing peak activities from a couple of weeks' process down to a few days. Program development in general was a slow process, measured sometimes in years, and there were fewer commercial packages available to meet the needs of the users. There were also fewer users who had relatively undemanding expectations, at least partly because of the limited availability of cost-justifiable technology. In other words, the expected rate of change was slower by several orders of magnitude.

Today's computing environment, even in a relatively small corporation, is bordering chaotic with hundreds of individual users with diverse demands for MIS services. Today's applications attempt to invent and facilitate new processes, rather than automate the old. It is also no longer uncommon to have MIS users from outside of the business organization--the real customers. MIS organizations are rapidly changing from the sole source of internal computing facilities into full-service organizations. In the process, they are finding themselves often competing with any number of outside sources to provide computing capabilities and services for their users.

From the business perspective, the emphasis is no longer in administrative systems, as much as it is in supporting the 'front-line' business. The fight for a competitive advantage has developed into a high-tech warfare. Business management is demanding timely and swift MIS support, ranging from consistently high levels of quality in existing services to a capability of handling sudden changes in demands for new services.

It appears to be easier, more justifiable, and desirable than ever to follow the leading edge of technology with a bewildering number of alternative solutions for any given MIS problem. Coupled with an increasing rate of change and the demands for a higher level of MIS participation in business, the volume of alternatives puts serious pressure on MIS planning skills and capabilities.

2.0 DEFINING THE PROBLEMS

A substantial number of alternative tools exist today in the area of MIS planning. They range from long-term, on-going strategic planning consultancy services to extremely sophisticated modeling tools specializing in particular software subsystems or applications and, in some cases, specializing in single device types.

There appears to be no shortage of tools for aspiring MIS Capacity Managers at technical levels. So what's the problem? Why is it then, that the MIS plans often fail to identify latent demand, major organizational and business changes, and even relatively slowly changing directions in technology? Most of the installations can relate to specific instances of deployment of new computing equipment, applications or services, which resulted in one of two verdicts: "Too little - too late!" or "Overkill!".

There are at least three major contributors that lead to planning difficulties which are often not addressed at all:

1. MIS attempts a corporate wide plan single- handedly.

It has been suggested by R. Nolan that the evolution of the planning function in an organization follows through cycles, comparable to the now classic 'Nolan's Curve'. The maturity level of an organization can be seen from the positioning of its planning function. In simplistic terms the first four phases can be characterized as follows:

At stage one, MIS planning is concentrated on an individual and is short-term in nature.

At stage two, a task force is set up on demand to formulate the MIS plan.

At stage three, a permanent IS Steering Committee plays a participatory role in MIS planning.

At stage four, a Chief Information Officer (CIO) function is filled and given the means and the authority to effect corporate-wide participation in MIS planning, potentially leading towards its full integration into business planning.

This type of model is useful for describing the current status of MIS planning in your organization and for defining the practical boundaries for a given planning exercise. It emphasizes rather than minimizes the planning problems which increase downwards and decrease upwards through the stages.

At lower stages, the inaccessibility to business planning information, the projected image of a task oriented MIS, and the prevailing communication gap between MIS and the business organization form major obstacles to effective MIS planning.

2. Planning is addressed only after a problem has surfaced.

To a large extent, this situation is more an indication of MIS planning taking place around the annual budgeting cycle, use of a too short planning horizon, or the remoteness of the MIS planning process from day-to- day operations, rather than an indication of merely insufficient planning.

Budget time rarely offers opportunities to gather the right people to participate in planning and even if it does happen, the plan tends to be put together in a hurry, without the full attention and concentration it requires.

3. The plan addresses a specific short-term problem in isolation.

In the absence of long-term plans, mapping MIS goals, strategies and their relationship to current business goals and priorities and attempts to justify any incremental upgrades to existing resources become major undertakings. Fighting for one short-term upgrade proposal after another also gives an often erroneous impression that business management is not receptive to MIS suggestions for change, and projects a less than professional image of MIS planning to business management.

As the symptoms described above suggest, not all of the capacity planning problems are technical in nature at all, but rather; they are management problems often caused and reinforced by limiting functional responsibilities and priority setting within the boundaries of organizational reporting lines.

Communication and interaction across the reporting lines flows much better in some organizations than it does in others. The former group also appears to be doing a better job in managing the gap and in encouraging cooperation between the MIS and business functions. Observed from a Capacity Manager's point of view, the latter group displays classical problems which require a strong matrix management approach, which generally speaking, tends to fall outside the MIS Capacity Manager's "comfort-zone" and area of expertise.

"Stage 1" companies with unrestricted flow of communication, manage to do an excellent job in MIS and capacity planning precisely for this reason. As a result, they may not have a pressing need to "mature" and appoint a CIO to bring the business and MIS closer, nor are they in any hurry to become a "Stage 4" organization. On the other hand, CIO's in some "Stage 4" companies are having a very rough time attempting to bring the right people to work together across the functional boundaries of their organizations and surmounting some of those historical barriers.

3.0 IDENTIFYING THE MIS PLANNING COMPONENTS

Figure 1 illustrates the first cut of the major components of MIS planning relative to the three levels of business plans: the strategic, tactical and operational plans. It shows that MIS needs to consider the three levels of planning for each of the business planning levels for each of the supported business function. The total number of planning components can get fairly large in a large and diversified company marketing numerous products or services.

The second cut is shown in **Figure 2** where one of the components from Figure 1 is expanded and further refined into following four components:

1. Business assumptions
2. Organizational assumptions
3. Technology assumptions
4. MIS support strategy assumptions.

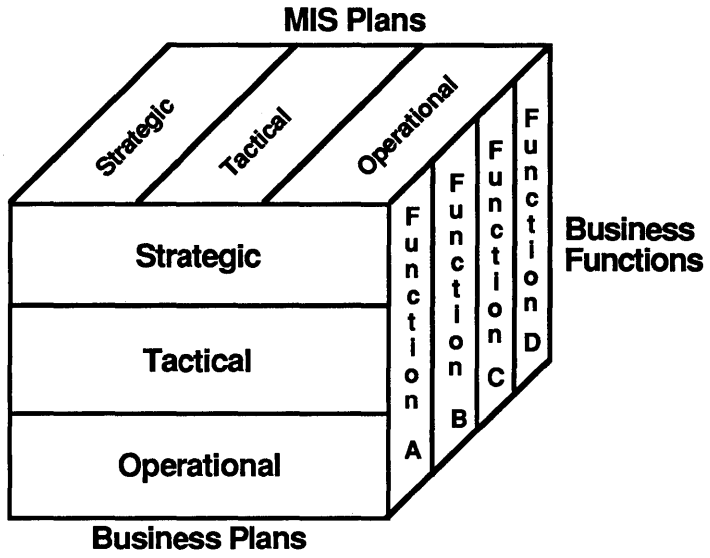


Figure 1: Matching Plans and Business Functions

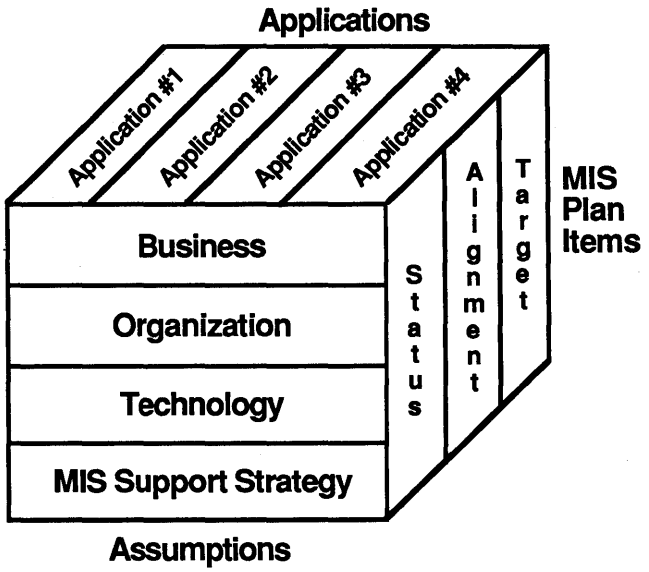


Figure 2: Exploded View

Each of the above then needs to be covered against the three position levels:

1. MIS position today in support of this business function
2. MIS alignment with the future position of this business function
3. Future potential of exploiting anticipated changes in technology in support of this business function.

In other words: Where do we stand today, where do we want to be tomorrow, and how are we going to get there?

Each of the four groups of assumptions may be refined in further detail, but that can be ignored for the moment. It is scary enough, as shown by the two illustrations above, that in order to produce a well thought-out MIS plan, over 300 areas need to be considered in some detail, and over 100 areas need to be considered even for a relatively short-term plan.

4.0 ALIGNING THE OBJECTIVES

The point is that the MIS plan is no longer an isolated document that defines the short-term needs for additional MIS resources. It is also almost impossible to successfully complete the plan without active participation from the business organization. The speedy acceptance of any MIS plan by business management is dictated by the degree it supports the overall business plans and activities.

A capacity plan is by definition a subset of an overall MIS plan. The other planning components also have to be completed to ensure that the capacity plan does not deviate from the stated direction and it gets accepted on merit, rather than on blind faith.

There is ample evidence, at almost every installation, that simply modeling the expected future growth of the existing workload is likely to result in a critical shortage of MIS resources to the short-term detriment of the company's primary business.

Planning in crisis mode to remove an already existing capacity problem is also highly undesirable. In crisis mode, the plan is typically prepared by one or two individuals, often resulting in a very limited one, which may be lacking long-term direction and synergy with any business plans.

A well-executed MIS plan requires the width of the company's business representation, a substantial amount of ongoing communication between the business and MIS organization, as well as the depth of MIS skills in capacity planning techniques.

Ideally, the Critical Success Factors of an organization should be structured in a fashion that they get addressed when the sub-organizations have addressed their CSFs, and so on, throughout the hierarchy, down to the personal levels.

Given that the consistent MIS service levels focus on achieving some of the critical goals outside MIS, it is necessary that the potentially affected business side of the organization participates in MIS planning. Firstly, in order to identify the shared objectives, and secondly, to gain maximum support and leverage to MIS plans.

Some MIS organizations are doing an excellent job of "selling" their services to their potential user groups. Some MIS organizations literally have a full-time sales person or a PR person on their payroll. While it may be an extreme approach and beyond the reach of an average MIS organization, this type of action does strongly promote joint ownership of problems, plans and justification for solutions.

MIS plan which used to have limited appeal to top management is transformed into a plan for the business organization to meet certain objectives. The chances of getting the MIS plan approved are greatly enhanced by a strong backing from the user community.

5.0 MANAGING THE PLANNING PROCESS

Each manager in any organization has at least a hidden objective to perform some planning. As a manager, one is expected to align the activities of his or her reports to support the "big picture", the business goals, and the goals of his or her immediate management. In addition to planning for future levels of compensation, managers are also responsible for planning for facilities.

How do these facilities differ from facilities and capabilities provided by MIS? How does planning for availability of office equipment or PCs differ from planning for different services and service levels provided by MIS?

There are some differences:

- Complexity of issues
- Uncertainty of future requirements
- Dollar amounts involved
- Cost allocation difficulties for new services to the first few users
- Traditional communications gap.

The biggest difference is that business management is rarely expected to plan for MIS related items, and the activities that would facilitate the planning done by MIS are not a formal part of their job description.

6.0 TAKING ACTION

Structured Planning refers to an "ideal" organization where a relatively small number of individual MIS-related planning items are constantly reviewed, updated and fed back to MIS as a part of ongoing responsibilities of several levels of operational business management across the organization. MIS' responsibility, beyond providing basic guidance and a mechanism to collect and analyse this information, should be to regularly review and summarize the information into a coherent MIS Plan which has the full support of the business organization.

"Stage 4" companies are addressing this need to cross the organizational boundaries from the top down, appointing a CIO function with a responsibility to make it happen. But, there are really very few reasons why the process could not be started from the bottom up, following the above guidelines. Maintaining informal networks of business associates is something each one of us does every day.

It is a relatively simple process to define the type of input that MIS management or the capacity manager needs from the business organization to do a good job. The difficult part is to define and implement the process throughout the company, which ensures that the assumptions and decisions made outside MIS filter back to MIS in a timely fashion. Doing it gradually on a personal level, rather than taking an ambitious wholesale approach right from the start, often has substantially better chances of bringing success on which to build later.

7.0 SUMMARY

MIS does have a lot to offer to the rest of the company in terms of the use of technology. And the business departments do add clout to the MIS plan as well as having the means to enhance MIS' understanding of the primary business of the company. What is needed is some basic 'marketing' action by MIS and some personal effort to promote a closer working relationship with business managers.

However contradictory it sounds, a successful implementation of structured planning demands flexibility in approach rather than a formalized process structure. Where "Stage 4" companies are tackling the cross-hierarchy communication problem by a top down management approach, the implementation of structured planning from the bottom up requires different a approach which relies on personal initiative and communication. The challenge is to gain the cooperation of key MIS users. Capacity plans which are clearly associated with the business demands, greater accuracy of the plans, easier risk management and cost justification provide the pay-off.

Who Goes There?
A Dialogue of Questions and Answers About Benign Hacking*

David F. Stevens
Lawrence Berkeley Laboratory

ABSTRACT

On August 23, 1986, it was noticed that the accounting files for one of LBL's computing systems failed to balance. On August 24, we received word that an unauthorized person was attempting entry into a Defense Department computer from LBL. Preliminary investigation indicated that LBL was the victim of a benign hacker, where "benign" is used in the medical sense. It was thought that the perpetrator was a graduate student from a neighboring university, and that it would provide a useful object lesson to other such folk if he were caught and admonished. LBL therefore embarked upon a journey of detection and containment instead of prevention. That journey continues today, having led first across the country, then across the Atlantic. In the course of the journey we have gathered a number of observations that should be of interest to anyone running a computer with any connection to the outside world.

Why the Computer Measurement Group?

The first and most obvious question is "Why should this material be of interest to CMG?" There are two reasons. The first is that one element of successful performance management is ensuring that enough capacity is available to satisfy the legitimate demands of all authorized users. Every cycle stolen or file hidden by an unauthorized user reduces the resources available for legitimate use. The elimination of unauthorized use thus extends the useful life of your present configuration.

The second reason for CMG to take an interest is that some of the tools developed for measurement and management purposes can also be used to detect unauthorized access. You are in the business of knowing where demand and usage originate; it is but a small extra step to distinguish between authorized and unauthorized usages.

What do you mean by "benign hacking"?

"Hacking" has gone through two metamorphoses of meaning. Its original meaning in the computing community was similar to its meaning in the golfing community. A hacker was one who butchered code. Then, some years ago, it was transformed from a term of opprobrium into one that indicated long hours willingly spent probing the dim recesses of system internals, searching for ways to accomplish Good Things. Finally, in the last decade or so, it has come to signify computing wizardry employed for personal purposes, frequently explicitly directed against governmental and large corporate installations.

In this dialogue I consider any activity resulting from (attempted or successful) unauthorized access to be hacking. (It is worth noting in passing that access by an authorized user at greater than authorized levels of privilege or for unauthorized purposes is unauthorized, and therefore a form of hacking.) Further, I define hacking that is limited to entry and consumption of resources, as distinct from intending to destroy or alter data, to be "benign". As noted in the abstract, *benign* is used in its medical sense: Not actively inimical, perhaps, but certainly intrusive and possibly painful. The presence of a benign hacker, like that of a benign tumor, is not to be considered a sign of good health.

Why limit the discussion to benign hackers?

Primarily because that's the kind I'm best qualified to discuss, but there are other good reasons:

- Benign hacking is more prevalent than the malignant sort (we think).
- It is more difficult to get people concerned about "harmless" hacking, even though in some respects the damage done to the institution can be as great for benign hacking as for hacking for gain. (Your image can suffer more from imaginative media accounts of recreational hacking than from serious accounts of hacker theft.)
- It is more difficult to detect benign hacking than vandalism.
- If you can detect and prevent benign hacking you have a good start at detecting and preventing vandalism and hacking for gain.

What do benign hackers do?

The total range of hacking activities runs from access only, just to prove it could be done (simply collecting scalps, as it were), through theft of resources (for the hacker's pleasure or do the hacker's personal work on someone else's computer), to grand larceny (altering bank balances or generating false invoices). We are concerned here only with benign hacking, which is essentially recreational in nature. In this context, most of the activity you are likely to encounter is generally related to snooping.

- Benign hackers look around; they browse through all directories they can gain access to, looking for interesting files.
- They document your commands (via trial and error; "HELP" facilities are a great help to them in this regard).
- They make lists of authorized users (the more valid ID's they know the more likely they are to get additional privilege).
- They read everybody's electronic mail (they are especially interested in messages about security breaches and countermeasures; they also look for passwords written in files).
- They look for programs that are looking for them.
- They look for (and try to crack) the password files.

* This work was supported by the U.S. Department of Energy under contract No. DE-AC03-76SF00098.

More active pursuits, but in the same general vein (that is, oriented towards allowing themselves undetected free use of your resources), are

- Attempting to disable accounting for their jobs;
- Stealing stale or unused accounts (usually by changing the password);
- Hiding files;
- Setting up bulletin boards;
- Attempting to disable security alarms;
- Playing games;
- Connecting to other systems from yours (and if you have a dial-out modem, they will try to use that, while you pay the phone bill).

How do they get in in the first place?

There are many avenues, ranging from the direct (someone gives them a valid password/ID combination) to the brute force (random dialing followed by random probes), but most of them take advantage of your cooperation through:

- Retention of default or generic logins with their original passwords;
- Use of common "temporary" ID's (such as *test* or *demo* or *guest*) with trivial passwords;
- Use of passwords identical to or derivable from ID's;
- Use of null passwords;
- Use of the same password/ID combination on multiple systems;
- Writing a password/ID combination for system A in a file on system B;
- Use of an unprotected remote login or file transfer from one system to another;
- The existence of commands (*who*, for instance) that can be executed without logging in.

They also run an extensive grapevine, and pass successful password/ID combinations around, both directly and through hacker-oriented nationwide bulletin boards.

How long does it take to break into a system?

That depends, of course, upon how much the hacker knows, how good your protection is, and how well your users follow good password practice. If you still use the default system operator or maintenance passwords that come with the system when it is is delivered, it takes about 10 seconds (for those are the first things tried). If you are well protected, and good practices are observed by all your users, it will generally take too long for casual hacking. Remember, however, that if another installation has network access to your system, you are only as secure as *that* installation and its user community. (This is a larger community than you might at first think. Widespead network access is now available to students at most of the institutions of higher learning in North America and Western Europe, plus many of the high schools, as well.)

Don't they have to know what kind of system it is before they start?

Most systems are very helpful to anyone who achieves a connection, and the login prompt often tells the caller the name of the company as well as the system and version number. Others are easily identified by the style of response to unsuccessful login attempts.

How many commercial sites are connected to networks?

Probably most of them of a size large enough to send someone to CMG. The telephone system, after all, is a worldwide network (and has been used to support transcontinental and transatlantic hacking). And any system with remote diagnostic capabilities has network access. Many service engineers have occasion to access many different systems, with differing password/ID combinations. It is fairly natural that they tend to record these nuggets of hackery on the fly leaves and back covers of their maintenance manuals, where they can be discovered through trashing. (In the '60's the term "trashing" referred to the intentional destruction of property. It now refers to a favorite "phreak"/hacker activity: the careful sifting of selected trash barrels looking for access-oriented jottings. It is successful often enough to encourage dedicated hackers and phreaks to fly across the country to raid a particularly fruitful site.) The point being, of course, that you are at risk not only from your own site, but from the engineer's trash bins in Boulder or White Plains or Dallas, as well.

But our dial-in numbers are unlisted....

Random dialers don't read directories, they just start dialing and hanging up if they don't get a modem tone. Many set their systems up to dial sequential numbers automatically for hours at a time while they are off doing more interesting things. And once they get a tone, as often as not, they announce the number (through a bulletin board or in the magazine *2600*) even if they haven't yet figured out what kind of a computer is on the other end. Some systems provide assistance, too: they have very helpful dialout programs that tell the user the previous number dialed.

Why should we care? (We don't have any secrets on our system.)

(This question tends to arise more often in the academic and research areas than in the commercial sector, but it has been heard there, as well.) You have data that is difficult or expensive to replace. You have data that contributes to strategic planning, to legal reporting requirements, to inventory management. Many of you have data that your employees and customers consider to be confidential. Some of you also have controllers who already think you're spending too much on computing, and who would not be happy to learn that some of it was going to unauthorized use.

But is this a real threat? Can anybody really get in this way?

Lawrence Berkeley Laboratory has observed a single hacker, over a four-month period, make more than 400 one-minute attempts to break into other systems from our site. Using a *very* limited repertoire of techniques (it's difficult to be truly creative in such a short time) he:

- reached about half the sites (i.e. was invited to try to login);
- succeeded in logging into 25 systems;
- became system manager at 9 sites!

That's better than a 2% success ratio at becoming system manager; more than three times that for simple access. That means that several of you here today are at risk. The odds today are, certainly, that if the hacker community wants access to your site badly enough, they *can* gain entry.

How did you find this guy? Why didn't you shut him out?

We found him through a combination of events, one of which was an imbalance in a double-entry bookkeeping system maintained for chargeback purposes. The discovery of the imbalance coincided with a report from another installation that an unauthorized remote entry had been attempted from Lawrence Berkeley Laboratory.

Our first thought was that the culprit was a student from a nearby college, so we set out to catch him in the act and create an example, rather than shut him out. In so doing we discovered much of the content of this paper. ("We", by the way, is primarily Dr. Clifford Stoll of our staff, assisted by folks from a number of affected institutions and law-enforcement bodies; chief among these outside folks was Steve White at Tymnet.) We *did* keep a rather close watch on our friend's activities -- for we had no assurance that they would continue to be relatively harmless -- and warned other sites whenever he succeeded in gaining entry.

One of the things we learned from having watched him for a while is that had we tried to shut him out when we first discovered him we would probably have failed, for he had already stolen several accounts. Thus, once we learned that it was not just a local student hack, we thought it worth pursuing to the end in the hopes of learning more.

How can one be sure one is safe?

If you are connected to a network or have a dial-in port you can be reasonably sure you're *not* safe. Collusion with your own staff aside, however, you can be as safe as you want to be. Safety consists in knowing all your users and in knowing the ownership of all files in the system. "Knowing your users" includes not only knowing who they are (a process of authorization and validation), but also of verifying that they are, indeed, who they say they are (authentication) every time they attempt to access the system. This is the fundamental purpose of a password scheme.

How much should we spend on protection?

This is a question of insurance, and as with all such questions you should weigh the cost against the expected value of the loss, and not overspend. Some of the costs on each side of the balance are relatively non-obvious. The cost of protection includes not only the cost of acquisition and implementation of protective measures, but also the costs inherent in the education of the users and in the irritation they experience at having to retrain their fingers to the new passwords. In estimating the expected value of a loss you must also include the cost of cleaning up after a breach. Even a "harmless" breach involves staff time and effort in detection, elimination of unauthorized files and acounts, tracing and prosecution (if warranted), and elimination of the system or procedural flaw that allowed the breach. These costs can be considerable. (In our case, they amounted to at least three months' full-time effort of a senior analyst, plus assorted bits and pieces of management time.)

What did you learn from your experience?

- When it starts, you never know how big it's going to get, or how far it's going to reach. We thought it would reach about a mile, and it turned out to extend from Berkeley to Europe. The level of interconnection of the world today makes it easy for a hacker to reach you in several hops, and he may choose to do so either to conceal his real origin, or because each step is limited for some technical or financial reason.
- The further it reaches, the more important it is to have a well-defined procedure to follow, and the more difficult it is to achieve coordination. The telephone company, for instance, prefers to deal with law-enforcement people in matters of call-tracing, rather than with technical people. So does the FBI. It has been difficult to get real cooperation from some agencies when contact is at the technical level, but technical contact is essential so that the sites being hacked can pool their knowledge of the hacker.
- Have clear communication paths. Know whom the telephone company will listen to, whom to call in any network to which you are connected, whom to call at the FBI or your local law enforcement agencies, etc.
- TYMNET has the tools and the willingness to help. Once alerted, they can complete a trace in minutes.
- The telephone company won't willingly trace beyond state boundaries without a federal warrant.
- Inform any other victims as soon as you can. Not only is this good citizenship, it will encourage them to notify you when the shoe is on the other foot.
- People use poor password practice -- even those who ought to know better. Among the poor practices exploited by our hacker are: continued use of default passwords; password identical to ID; trivial password (i.e., easily guessable, such as name when ID is initials, location, system name, same password on several systems); existence of non-password-protected ID's (e.g., *who* on some *unix** systems); listing of ID and associated password in the clear in files with suggestive names.
- Passwords decay with time. Other people get to know them through several mechanisms (accident, gift, guesswork, ...).

---

* *unix* is a registered trademark of AT&T Bell Laboratories.

- Whenever people are required to use system-generated passwords, they will write them down where a hacker may find them.
- Be careful when demonstrating systems to unknown people (at least one hacker claims to have gotten a password by looking over a salesman's shoulder during a demonstration).
- It's more difficult to discover unauthorized use *after* a valid account has been stolen than before.
- Don't use E-mail to discuss any aspect of a hack, unless you want the hacker to read it, too.
- An attached printer is an indispensable tool for serious tracing. Without one it is generally impossible to reconstruct the hacker's session. (If he has system privilege, he can destroy the log of his session before you can read it.) A printer linked in parallel with his output gives you a real-time track that he cannot erase.
- A network is weaker than its weakest link. (And all of the networks we belong to have some *extremely* weak links.)

What can we do to detect benign hacking and to protect against it?

The following lists are suggestive rather than exhaustive. No attempt is made to rank the items in order of effectiveness or cost; individual situations are too different to make such an attempt meaningful.

To detect:

- Look for hidden files. Many systems have legitimate ways to "hide" utility files so that they don't clutter up working directories. In *unix*, for instance, any file name beginning with a dot (such as *.login*) is hidden. It is also possible to place files into directories that are not usually searched.
- Look for files with "unauthorized" contents. (Since different organizations have different rules about off-hour use of computers, the definition of "unauthorized" can vary widely.) If you employ this procedure, it is important to ensure that the looker is ethical, and does not divulge the contents of any of the files looked at.
- Require that every network access carry trace-back information so that you (or the network operator) can determine the source of all traffic. Review the log from time to time and look for anomalies.
- Provide your users with enough accounting information so that they can discover if anyone else is using their accounts.
- Institute chargeback, so that they have a reason to read the accounting information and to inform you if it seems to be out of line.
- Employ double-entry bookkeeping. (One way is to keep a write-only record of all resource usage and then to compare that to all "charged" usage.)
- Tell users of unsuccessful entry attempts in their name.
- Tell users the date and time of their last successful login.
- Encourage the users to read those messages and notify you of any anomalies.

To protect:

- Authorize users individually and personally. Verify the user's identity (and employer) before granting access.
- Require the use of passwords for access, and adopt a sensible password policy. Some possibilities are:
  - -- Password cannot equal login name.
  - -- Changes of password cannot equal previous passwords.
  - -- Password must be at least 6 characters long, and contain at least one non-alphabetic character.
  - -- Password cannot be all the same character.
  - -- Password cannot contain a recognizable part of the login name.
  - -- Password must expire within 180 days of being issued.
  - -- Passwords belonging to terminated employees should be terminated.
- Require password changes. One practitioner has said "Treat your password like your toothbrush: Change it once in a while, and don't pass it around."
- Control access to particularly sensitive data by location (terminal or port of origin) as well as by name (password/ID).
- Let login ID's expire (to avoid proliferation of stale accounts).
  - -- Initial logins should be deactivated very quickly if they are not used.
  - -- "Active" logins should expire after some reasonable period of non-use.
  - -- A login should be terminated as soon as any employee who knows it leaves the company (for any reason).
- Fix any known loopholes that allow someone to increase his level of privilege.
- Protect network access.
  - -- Avoid network access procedures that require or encourage the writing of passwords in files.
  - -- Limit the privileges of any process that originates remotely.
  - -- Monitor the use of remote logins.
  - -- Install password protection on network access ports, including those used for remote maintenance.
  - -- Adopt a dial-back verification procedure for remote telephone access.

-- Restrict physical access to workstations possessing automatic login procedures for access to your mainframes.

- Protect log and journal files that give the history of interactive sessions. These can be read, and they can be as useful to a hacker as ordinary text files.
- Set file access protection defaults to something more restrictive than *world*.
- Require your system to remain anonymous until *after* you get a valid login. (Many systems provide a great deal of help by welcoming login attempts, or even when responding to a modem call. Don't provide anything other than bare ID and password prompts until you know who's at the other end.)

What do we do when we find someone?

That's up to you, of course. That depends upon what's at risk, what has been damaged, and how much effort you are willing to devote to the chase and possible prosecution of the offender.

Do you have any last words?

Only the following warning, adapted from Tolkien:

> It does not do to leave a live
> dragon out of your calculations
> if he has network access to your
> computer.

REAL WORLD PERFORMANCE MEASUREMENTS FOR UNIX-BASED SYSTEMS

Neal Nelson
Neal Nelson & Associates

ABSTRACT

During the past three years, benchmarks on multitasking Unix-based machines have decisively demonstrated that the processing capacity must be measured to determine comparisons between machines. Whether they are $5,000 micros or million dollar mainframes, the benchmarks underlined the unreliable nature of MIPS, Whetstones, milliseconds, megahertz or any other commonly quoted statistics. This paper reports on these benchmarks.

BACKGROUND

Before explaining my benchmarking activity and presenting some of my findings I would like to explain the sequence of events which prompted me to start working in the performance measurement area.

I started a software house in Chicago in 1973. For eight years my firm installed primarily small accounting systems, mostly on IBM System/32's, 34's and Series/1's. We also installed several non-IBM systems like Wang, General Automation, DEC, Hewlett Packard, Qantel, Nixdorf, and so forth.

We did primarily custom installations, and by 1982 we had written the entire range of business applications on twelve different brands of computers, in six languages, under sixteen operating systems. I had grown tired of "re-inventing the wheel", and so I looked for a market where I would not be subject to a new language and operating system every time a new product family was announced.

After studying the possibilities, I decided to move into the Unix marketplace and develop a package of general accounting programs in the "C" programming language. By 1984 we had substantially completed work on the accounting programs and had several successful installations.

Up until this time, the Unix machines we had been installing were the Onyx brand computers. We felt that we should seek another brand to recommend to our customers as the offerings from Onyx at that time were not competitive.

I knew nothing about benchmarking, and asked a business associate for information. I was provided with a set of public domain benchmarks, some sample numbers from previous tests, and instruction on how to run the benchmarks. I then went out and began to test every machine I could get my hands on.

Based on these tests results, I recommended a computer to one of my prospects. When my application software was installed on the recommended machine, it ran VERY slowly. At first I thought that the machine might be defective, since the benchmark had indicated that the machine would be quite fast. I ran the benchmark on the customer's machine, and the benchmark ran like lightning but my accounting programs just crawled.

At this point I knew I had a problem. I began to analyze the benchmark and realized that it was measuring shell scripts, compiles, calculation speed and so forth, but it was not measuring what most accounting programs do most of the time which is disk I/O (particularly random disk I/O).

I did an analysis of my accounting programs and broke the code structures down into fifteen broad categories of functionality (16 bit integer math, 32 bit integer math, sequential disk I/O, random disk I/O, etc.). I created my own benchmark from these function groups.

Running this benchmark on my old Onyxs and on the new machine I found that in several areas the new machine was faster, but in two or three it was slower, and sometimes a lot slower.

This information told me how I had to modify my applications to eliminate all unnecessary uses of the slow techniques. When that was completed, the programs ran at an acceptable speed, and solved my client relations problem as well.

At this point I vowed to never subject my clients to hardware problems of that nature again. So, using my own benchmark suite, I began a series of tests on all of the machines. The results differed from the first set of tests. This concerned me because often the results appeared very similar on machines which experience had shown to be very different in power and speed.

The first version of my benchmark was a single user benchmark. Soon I converted the benchmark into a multitasking benchmark. Again I retested all of the machines. This time the results agreed with my experiences as well as with user reports. By measuring degradation under increasing levels of load, I was able to make fairly accurate predictions about how a computer would perform in the "real world".

That multitasking benchmark has gone through several generations, and has since become a product at my firm. The product has evolved into eighteen test categories covering over 30 functions. Over the past three years we have run this benchmark on over 200 Unix-based computers. We maintain a database of test results which allows us to compare "anything to anything" very quickly. From our database we can print eighteen-page reports which compare two machines using a graph and a bar chart format for each of the eighteen categories.

My mental attitude on computer performance measurement has evolved into a series of "What happens if" questions such as, "What happens if we increase the memory size?", or "What happens if we install a new release of the operating system?"

The following illustrations will show some of our findings. I cannot include all 18 graphs for each comparison, so I have selected two or three (usually a calculation and a disk intensive test) that demonstrate the point.

EXAMPLE 1: NEW RELEASE OF OPERATING SYSTEM

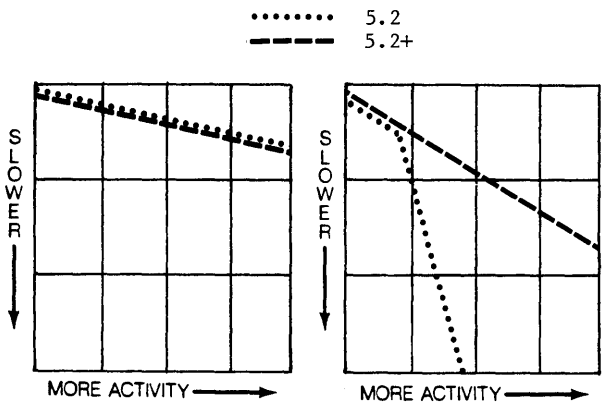
The first example deals with a new release of the operating system.

A manufacturer of a desk top Unix machine announced a new release of the operating system. The original release was 5.2 and the new release was sort of 5.2+.

The original release is represented by the dotted line and the new release by the dashed line.

The first graph of calculation type work shows little difference between the two versions.

Graph two, however, shows that for disk intensive tasks the original release experienced extreme degradation after four copies fo the test program were executed, whereas the new release seemed to correct the problem.



Graph 1 Graph 2

EXAMPLE 2: CLOCK SPEED COMPARISON

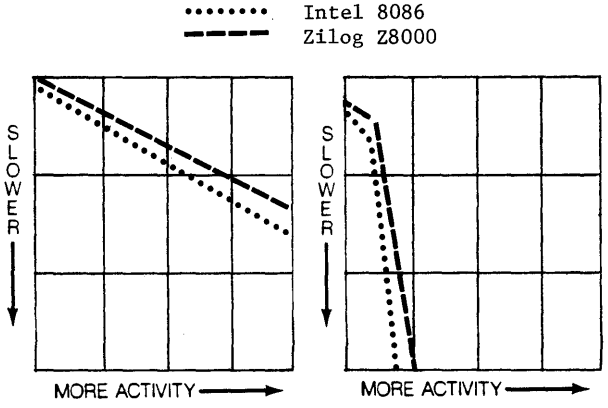
The next examples deal with different chip technology and clock speed.

Both of the next two examples were 16 bit computers. Both have the same memory size, and similar disks. The major difference is that one machine is based on a 4 megahertz Zilog Z8000 chip, while the other is based on a 10 megahertz Intel 8086 chip.

One would expect that the Intel based machine would be significantly faster than the Zilog based machine, however, graph three shows that the Zilog is actually running 20% faster that the Intel. This test consists of integer math, function calls, string manipulation, looping, branching, and so forth (Graph 3). There is no activity which would slant the test in favor of the Zilog chip.

Apparently, a Zilog Z8000 based computer with a 4 megahertz clock could do more useful work than an Intel 8086 based computer with a 10 megahertz clock.

Graph 4 shows a disk intensive test for the same two machines. Both machines degrade quite dramatically, however, the Zilog based machine is still a little faster than the Intel based machine.



Graph 3 Graph 4

EXAMPLE 3: 8086 vs. 80286

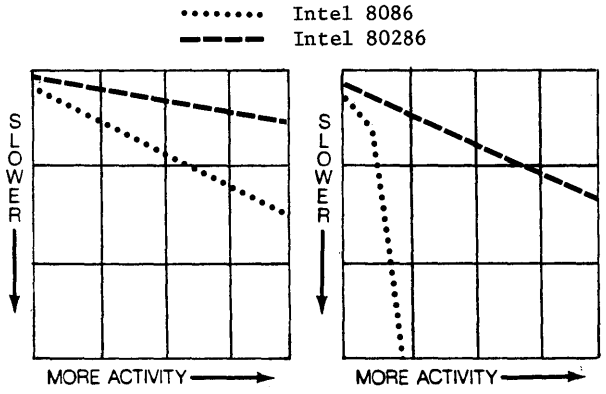
The next comparisons are for two Intel based machines.

One uses an Intel 8086 chip and the other uses an Intel 80286 chip.

Both machines ran Xenix, and both machines were manufactured by the same company. I felt the 80286 based machine was created as an "upgrade" product for the customers that already owned the 8086 based machine. I wanted to see how much improvement the customers might expect if they choose to upgrade.

Graph 5 shows calculation type work, and it is apparent that the 80286 based machine (represented by the dashed line) is running almost three times faster than the 8086 based machine.

Graph 6 shows disk intensive work, and in this case the improvement is even more dramatic.



Graph 5 Graph 6

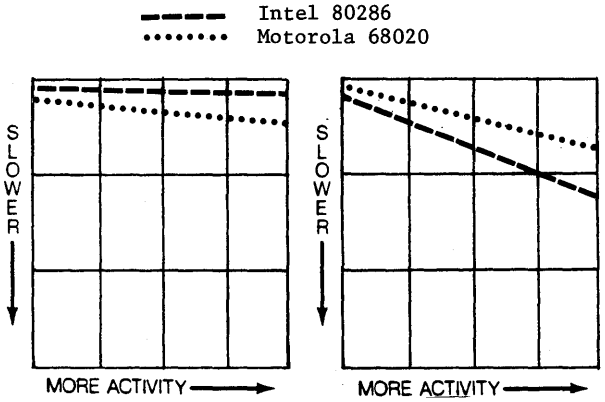
EXAMPLE 4: 80286 vs. 68020

Another interesting question is, "Who makes the most powerful chips?" For example: is a computer using an Intel 80286 faster than a computer using a Motorola 68020?

The next two graphs represent comparisons of machines which are the same in many respects. They are made by the same manufacturer, they have the same memory size, they use the same disk drive and disk controller.

Graph 7, however, shows that for calculation type work the 68020 based machine (represented by the dashed line) is dramatically faster.

Graph 8 shows that for disk I/O the relationship is reversed. The 80286 based machine is almost 25% faster.



Graph 7 Graph 8

These graphs illustrate my most basic conclusion after benchmarking over 100 computers: Don't assume anything. Test and measure everything, then draw a conclusion.

The 68020 is running with a 16.7 megahertz clock (more than twice the 8 megahertz of the 80286), and yet, calculation work is much more than twice as fast. They are both 32 bit chips. These machines are built with interchangeable parts for the backplane, disk controller, and disk drive. Why is the 80286 25% faster on disk I/O?

I don't know the answer, but it must be something to do with the efficiency of the compiler, the Unix port, the disk interface routines, and so forth.

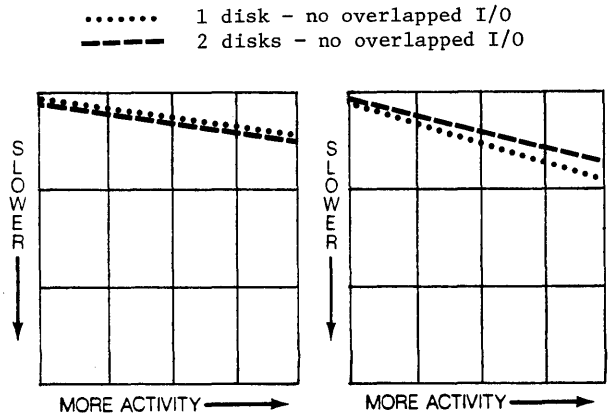
What I do know is that there are significant differences which contradict what would be expected based on clock speed, access time, etc. I have concluded that everything must be tested.

EXAMPLE 5: FILES ON MULTIPLE DISKS

One more question is, "What performance increase will result from spreading disk work files over two drives rather than having them all on one?"

Graph 9 is calculation type work on two systems. One system is configured with all work files on one disk, while the other has the work files evenly spread over two drives. Both lines are the same, which is correct, since this test does no disk I/O.

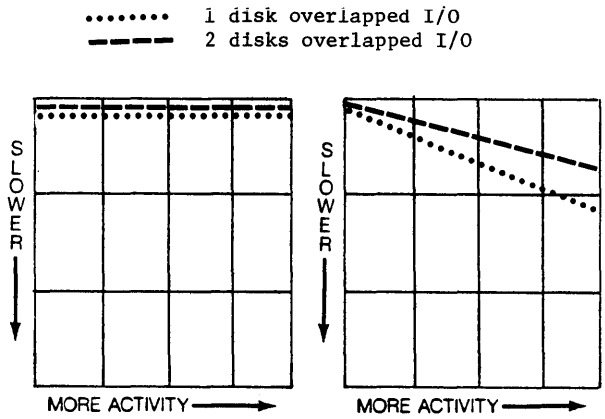
Graph 10 is the disk I/O intensive test, and we see a modest (15% to 20%) improvement. This is somewhat disappointing. Both disks are connected to the same controller. I called the manufacturer and learned that this machine does not do overlapped seeks. Hence the modest improvement.



Graph 9 Graph 10

Graph 11 shows calculation type work for a pair of 68020 based machines from the same manufacturer. These machines have the same disks and disk controller as the previous 80286 based tests. There is no difference in the graphs for calculation type work, which is expected.

Graph 12 shows a 30% to 40% improvement in disk throughput when the files are placed on two drives (still one controller). These are the same disks and same controller as in the previous example where there was a 15% improvement. I made another call to the manufacturer and learned that the controller is capable of overlapped seeks, and that the 68020 version of Unix from this company can support overlapped seeks, so on the 68020 machine they download different microcode to the controller and do overlapped seeks.



Graph 11 Graph 12

You might understand now why my rule is: "Benchmark first, ask questions later."

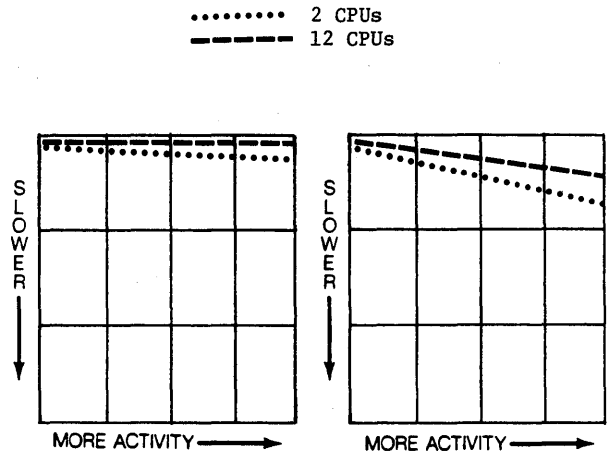
EXAMPLE 6: TIGHTLY COUPLED MULTI-CPU

How about one of these new tightly coupled multiple CPU computers where they put a dozen small CPU chips in a box and get them all working at the same time?

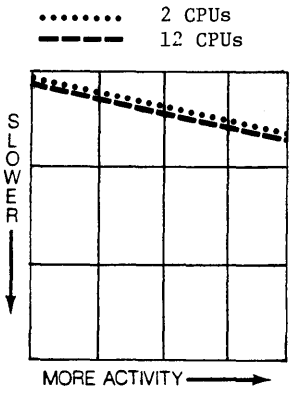
Graph 13 shows a comparison of two configurations for one of these multiple CPU machines. One configuration has 2 CPU's and the other has 12. Graph 13 represents calculation intensive work, and although the resolution of the graph doesn't show it, the 12 CPU machine shows almost zero degradation up to twelve copies, and at twenty copies it is running exactly 600% faster than the 2 CPU configuration. This is exactly what we would hope to find.

Graph 14 is a disk intensive test of the same two machines. It shows that the 12 CPU machine is twice as fast as the 2 CPU machine, but no longer six times as fast.

Graph 15 is a test where random seeks are performed, and this test is almost always limited by the seek time of the disk. The same two machines on this test show no improvement, which illustrates that if your application is bottlenecked on random disk I/O, adding more CPU's is not necessarily going to help.



Graph 13 Graph 14



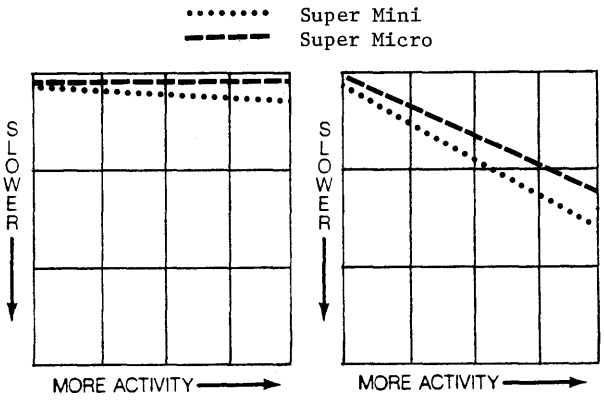
Graph 15

EXAMPLE 7: SUPER MINI VS. SUPER MICRO

O.K., so let's compare a $150,000 super mini to one of those $30,000 super micros which have been claiming to be faster at a fraction of the cost.

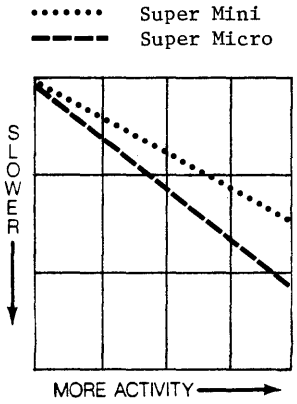
Graph 16 shows calculation type wrok where the dashed line is the micro and the dotted line is the mini, and indeed, the cheaper machine is actually running faster (quite a bit faster).

Graph 17 shows disk intensive work with an assortment of sequential and random I/O. The cheaper machine is still 25% or so faster.



Graph 16 Graph 17

Graph 18 is the random read test, and in this case the mini is faster (which apparently reflects the faster access time of the disks).



Graph 18

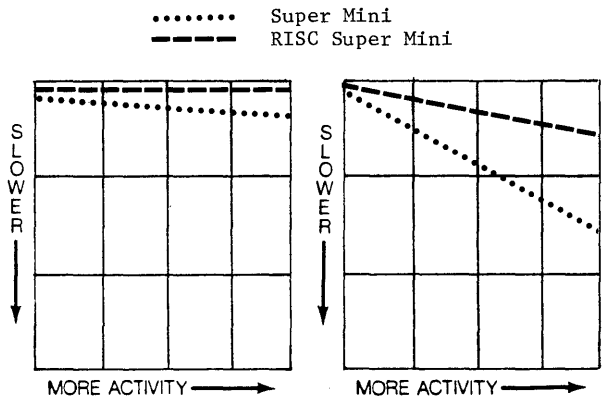
This was not a true "apples to apples" comparison because the mini had much larger disks and many more ports than the micro. But in terms of pure speed and price, it is sort of an eye opener. You could buy a lot of disks and some extra ports with that $120,000 difference in the prices.

EXAMPLE 8: SUPER MINI VS. "RISC" SUPER MINI

How about a test of a super mini which uses the RISC (reduced instruction set computer) architecture compared to a super mini which uses the conventional instruction set architecture. Is the RISC machine better?

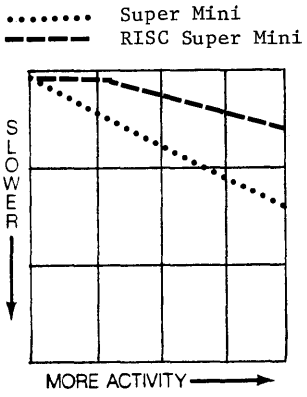
Graph 19 shows calculation type work for RISC versus Super Mini, and indeed, the RISC is faster by a wide margin.

Graph 20 shows disk intensive work, and again, the RISC machine is a lot faster.



Graph 19 Graph 20

Graph 21 is the random access test, and the RISC machine is over twice as fast.



Graph 21

This was a true "apple to apples" comparison. Both machines had comparable numbers of disks and ports. The memory size was different, but neither machine would begin paging during these tests, so that would not affect the results. In fact, both machines were running Fujitsu Eagles as disk drives with the same access time, so it is interesting that the RISC machine did so much better in random disk I/O. It is apparent that it is just a better machine.

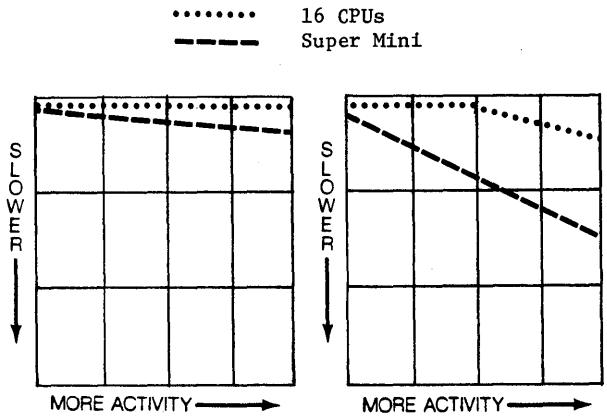
One interesting footnote is on the random test where the RISC machine showed almost no degradation up through the sixth copy. Upon investigation I learned that the RISC machine was configured with several megabytes of disk cache buffers in memory. Up until the sixth copy no disk I/O was actually done because all of the requested records were already in memory. It is only after the buffers were saturated that we actually saw the slope of the curve for the RISC machine.

EXAMPLE 9: MULTI CPU SUPER MICRO VS. SUPER MINI

The next three graphs show what happens when sixteen microprocessor chips in a tightly coupled machine gang up on a classical super mini.

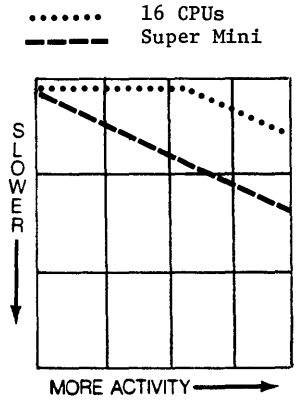
Graph 22 shows that the multi CPU machine is superior for calculation types of work.

Graph 23 shows that the multiprocessor is faster for disk intensive work, and again, we see the effect of very large disk cache buffers in memory.



Graph 22 Graph 23

Graph 24 shows the multiprocessor to be faster in this random access test as well.



Graph 24

It must be remembered, however, that multiprocessors may only be superior in cases where large numbers of users are doing different tasks. If the work load dictates that single users will run big jobs, all but one of the micro processors might sit idle while one of them struggles along. It is essential to think about the application and intended use of the computer before making a final choice.

CONCLUSION

The conclusion that we have come to, and the conclusion that we urge upon our clients, is to think about the intended use of the computer. Will it be single or multiuser, calculation, scientific, office automation, database, accounting, communications? Just what will be the normal use of the machine?

Then, find a benchmark which effectively measures that type of work and test each machine which seems interesting. Don't place faith in the clock speed, MIP rating, bus width, access time, cycle speed, memory size or anything else. Measure it with an appropriate benchmark, and there should be no suprises when the machine is placed in service.

STORAGE PERFORMANCE -- AN EIGHT YEAR OUTLOOK

Stephen Goldstein
International Business Machines Corporation
General Products Division, San Jose, California

ABSTRACT

This paper treats I/O subsystems in moderate to large commercial MVS installations from the perspective of performance requirements. It examines requirements over the past several years and projects them into the foreseeable future. Movitated by a 1985 customer study, the projections recognise two categories of data; "system data" which is commonly shared among all tasks and "user data" which is typically application specific. Having projected requirements, the paper suggests a device and attachment structure that would support these requirements.

1.0 INTRODUCTION

This paper treats I/O subsystems in moderate to large commercial MVS installations from the perspective of performance requirements. It examines these requirements over the past several years and attempts to project them into the foreseeable future. Three specific parameters are addressed: storage capacity, thruput capability, and response time. These parameters are tracked historically from 1980 to 1985 and then projected into the future with specific snapshots focused at 1990 and 1995. The basis of the projections is a combination of the historical trends observed and an allocation of those trends to fundamental causes -- "driving forces" that may either strengthen or weaken over the next several years.

Having made the projections, the obvious and interesting question is; "What will (or must) the technology be in order to support these requirements?" This question is addressed with some fairly simple performance modeling combined with some conjectures on where technology might be.

2.0 PROJECTIONS BASED ON CUSTOMER STUDY

As stated above, projections will be based on trends observed between 1980 and 1985. The 1985 picture is developed from an extensive customer study including measurements, traces, and system programmer interviews conducted in the spring and summer of that year. Ten commercial MVS installations are represented.

At each installation, measurements were taken at typical peak periods during the day, usually between 10 to 11 in the morning and 2 to 3 in the afternoon. These measurements included RMF (Resource Monitor Facility of MVS) and GTF SIO trace (the General Trace Facility's Start I/O trace). In addition, during off-shift, each installation cooperated by running a VTOC snooper program that collected detailed data set information for each volume in the I/O subsystem.

Without revealing what specific commercial installations were involved in this study, it is useful to at least generally identify the type and size of account and applications that were included.

Installation(s)	Application(s)
Aircraft Manufacturer	Manufacturing support
Integrated Oil Co (2)	Credit card processing Program development
Auto Manufacturer	Graphic design Program development
Heavy Equipment Manf (3)	Parts data base Program development
Life Insurance Co	Policies, claims and billing
Regional Telephone Co	Billing and inquiry
Regional Bank	Customer account files

These installations averaged 37 MIPS (millions of instructions per second) across one or more processors attached to an average of 140 GB (gigabytes) of DASD. For ease of comparison, an IBM 3090-200 is nominally 28 MIPS and a box of IBM 3380 single capacity storage contains four devices totaling roughly 2.5 GB of capacity. So the average system surveyed was the equivalent of 1.3 3090-200's with 56 boxes of single capacity 3380's attached.

Projections for both capacity in GB and I/O rate in accesses per second are normalized to the aggregate processing power of the installation. That is, GB/MIP and accesses/second per MIP are the parameters of interest; the data presented will support scaling to processor power.

2.1 Capacity

Figure 1 shows that an estimate of 3.7 GB per MIP fits quite well with data from the ten installations included in the 1985 Customer Study. Since a linear relationship exists the data supports the scaling of capacity to processing power. Historical data since 1971 indicates that while the large system GB per MIP ratio has been very nearly constant over the past several years, an extended period of growth occurred between 1972 and 1978. This historical trend is displayed in Figure 2 which represents a large sample of high-end computer installations including storage devices and processors from a number of vendors.

To forecast the relationship of GB to MIP over the next several years, it is useful to

contrast the period of the late 70's, when a growth in GB/MIP was occurring with the early 80's, a rather stable period (figure 2). Over the time period from 1972 to 1978 the GB/MIP value increased at 15% CGR (Compound Growth Rate). During those years, many installations converted their operations from batch applications that loaded data sequentially from tape and ran sorted, batched transactions against it to integrated on-line data bases processing randomly arriving transactions in real time. In essence, the method of real time transaction processing pioneered by the airlines in the 60's became standard practice for many installations in the 70's.

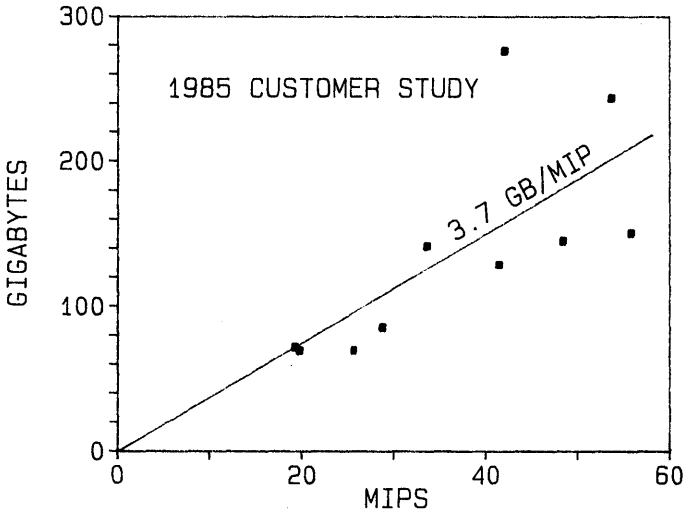


Figure 1. GIGABYTES PER MIP

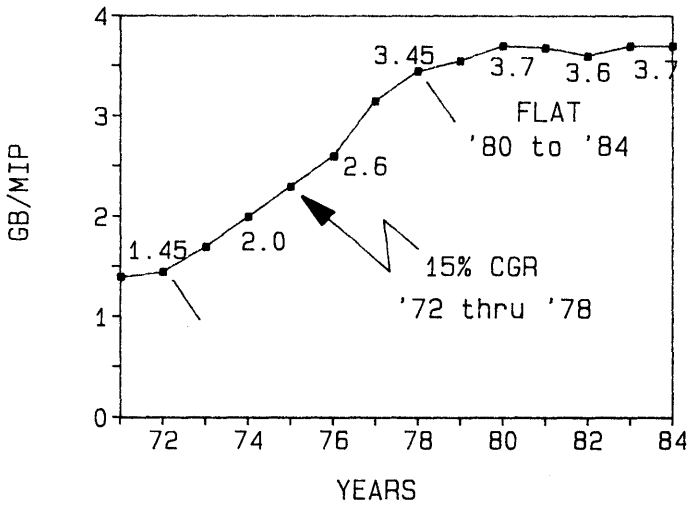


Figure 2. GB/MIP HISTORY

At the same time a technology revolution was occurring. Auxiliary storage devices changed from individually owned removable media (e.g., IBM 3330's) to system owned fixed media (e.g., IBM 3350's). Operating systems participated by introducing facilities to enable the use of fixed media as a repository of permanent data. These simultaneous changes in the nature of large system applications, storage technology, and control programs combined to support significant and extended growth in GB/MIP.

From the recent stability in GB/MIP, it would seem as though the ratio of GB to MIPS is likely to remain near 3.7 until some new forces come along to alter the relationship -- in the applications arena, a couple of changes are on the horizon. One change is the "office of the future" or paperless society where large volumes of inherently non-digital information are captured, digitized, stored, retrieved and displayed as needed. By its nature this type of information is relatively more demanding of storage space than processor power. Its incorporation into storage subsystems will almost certainly increase the GB/MIP ratio. Another application change with the potential to increase GB/MIP is the use of the large "glass house" system as a repository for information which is created and processed on personal computers and intelligent work stations. The impetus would be to take advantage of the economy of scale that persists in large capacity storage devices as well as the inherent efficiency of installation-wide backup and recovery. Backup and recovery is both cheaper and more reliable when done routinely and automatically across a large installation, rather than replicated across many individual users each with relatively slow and expensive backup devices.

On the technology side, complementary developments are pending. The cost of storage is dropping and may drop more dramatically if optical technology fulfills its promise. Even without introducing a new technology, the cost of storage may drop as cache supplies performance and frees magnetic DASD designers to optimize to lower dollar per MB design points without undue concern for performance bottlenecks.

GB per MIP projections

These forces portend a rise in GB/MIP, but that rise will be tempered by inertia. The rapid, extended 15% CGR in GB/MIP that occurred in the 70's operated on, by today's standards, a very small base. Recall that the state-of-the-art drive of the early 70's was a 100 MB 3330, roughly the capacity that many personal computers will have in the next couple of years. As a result, my projection is that GB/MIP growth will resume in 1988 and continue over the next several years at a modest CGR of 2% per year.

2.2 Access rate

Figure 3 represents data from the 1985 Customer Study and shows that an I/O access rate of 15 accesses/second per MIP fits the data quite well. Again a claim that access rate is linear with processor power is supported. Investigation was done on the one non-conforming point on the far right in Figure 3 to see if its installation had any unique characteristics. It did. While all the other installations were heavily into on-line real time processing, that individual installation primarily processed in batch mode and using extraordinarily large block sizes even during the measured prime shift peak period. A good history for access rates was not easily established. One customary complicating factor is the exclusion of paging from the I/O subsystem, as this paper does. In the late 70's and early 80's, paging was universally considered part of the normal I/O workload.

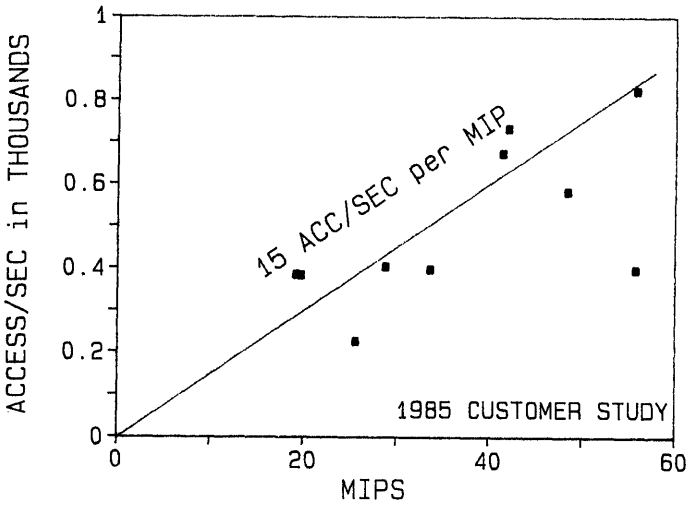


Figure 3. I/O ACCESS RATE PER MIP

The data here is based on an article in the IBM System Journal (1) and discussions with its author. The I/O access rate in 1980 for large systems is estimated at 25 access/second per MIP. The drop from 25 to 15 represents an annual rate of reduction of 10% over the 1980 to 1985 time period.

Several forces were almost certainly at work to fuel this dramatic reduction; they include more processing per I/O, exploitation of bigger processor memories, user friendly applications, and transaction processing. More processing per I/O is a phenomenon closely related to using larger block sizes on DASD and cost as a function of block size. Large block sizes allow more efficient use of DASD space and hence are desirable. Their use naturally increases the amount of data transferred per I/O operation and reduces the access rate per million instructions executed. Reinforcing the exploitation of large blocks is an argument from the cost of an access perspective. Transferring large blocks of data is efficient, because most of the device time expended in accessing a block is fixed overhead.

For example, in current technology, accessing a 4K byte block of data from a DASD devices results in roughly 20 ms of device busy time of which less than 1.4 ms is data transfer. Other elements of device busy time include seek, latency, search, RPS reconnect and protocol. They are, for the most part, independent of the amount of data transferred. If an 8K block (or two consecutive 4K blocks) is retrieved device busy time would rise from 20 ms to 21.3 ms, an increase of about 7% for twice the amount of data retrieved. If the additional 4K bytes retrieved is useful, this is indeed a bargain that should not be overlooked. The trend to larger blocks, and therefore more processing per block transferred, will continue although natural barriers, specifically the Law of Diminishing Returns, limits its growth.

Exploitation of larger main memories is a strong factor in the decrease in I/O rates per MIP. Over the past several years, operating systems and major applications have concentrated on this. For example, consider data base buffer pools. Data base applications have built elaborate lookaside buffer schemes to reduce the need for physical I/O accesses to storage devices. The dramatic fall in the cost of processor memory, driven by larger and larger capacity chips, is fueling the trend to utilize processor storage to reduce the need for accesses to auxiliary storage. It will probably continue and may intensify over the next several years.

At first glance it might not seem apparent that user friendly applications and transaction processing have been "drivers" of the reduction in storage access rates; they have. Both of these factors have worked in a similar way, by diverting a substantial amount of the processors power away from the main stream application and into activities that require few I/O operations. In the case of user friendliness, many instructions go into screen manipulation. Transaction processing diverts processor resource to network control and message processing. I forecast no change in the demand for user friendliness and on-line processing; however, the transition is largely done and as forces for change, they will have a diminished effect over the next several years.

I/O Access Rate Projection

To summarize the I/O access situation, exploitation of larger processor memories will strengthen, more processing per I/O trend will continue to reduce access rates, while effects of user friendly applications and on-line transaction processing will abate. In net, I/O access rates per million instructions executed, which fell 10% per year from '80 to '85, will continue to fall but at a slower annual rate of 7% per year.

2.3 Response time

In the 1985 Customer Study, response times varied widely from customer to customer and data pool to data pool with no particular correlation to the processing power of the installation. In Figure 4, there is one vertical bar for each installation; they are arranged according to increasing average response times. The upper end of the bar represents the highest average response time among the installation's data pools; correspondingly, the lower end on each bar represents the lowest average response time among that installations data pools. The length of the line is a measure of the variability of response time within the installation.

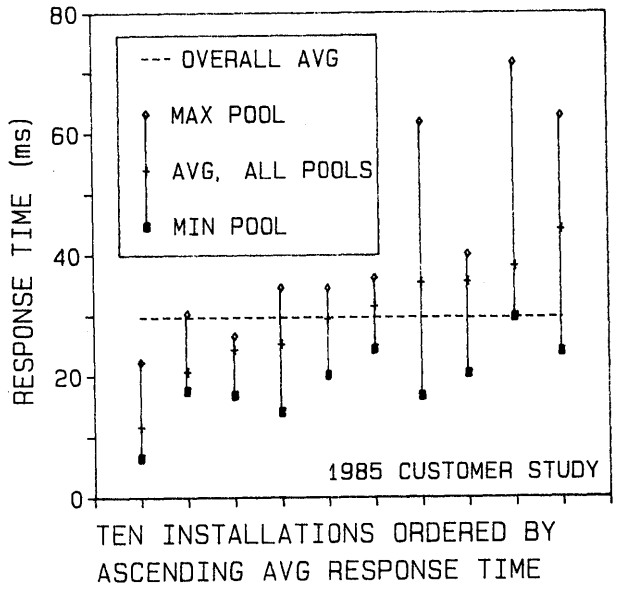


Figure 4. RESPONSE TIME RANGES

A couple of specifics are noteworthy. The installation represented by the left-most bar enjoys the lowest response times in all three categories, and is mostly cached. The next three installations all spend substantial resource in tuning and balancing their I/O subsystem and appear to achieve a measure of success in return for their efforts.

Looking back to 1980, the workhorse DASD device was an IBM 3350 and its compatible equivalents along with some IBM 3330 model II class devices. From my recollections and those of my colleagues, and from the review of a few previous internal studies, the typical response time in the peak busy period was in the range of 40 to 60 ms. In this paper, I have elected to use 46 ms as the 1980 representative value. This represents an annual rate of reduction of 9% when compared to the 30 ms average response time of the 1985 customer survey represented as a horizontal line in Figure 4.

The forces decreasing response times from the I/O subsystem are application, system, and technology based. From the application perspective, a desire to simultaneously increase transaction complexity and improve end user productivity supports the requirement for lower DASD response times. The system perspective is that high DASD response times require high levels of multiprogramming to utilize increasingly faster processors. But high levels of multiprogramming introduce their own inefficiencies, such as long queue searches and increased resource and lock contentions; they are to be avoided if possible. Fast DASD response times mitigate the need for higher levels of multiprogramming. From the technology perspective, lower DASD response times are achieved through the exploitation of electronic technologies to improve connectivity, speed data transmission between the spinning magnetic media and processor memory, as well as to provide high speed outboard RAM for caches and solid state devices.

Response Time Projection

The forces to reduce I/O response time will accelerate over the next several years. Particularly as data bases move to the increasingly popular relational approach, the pressure for faster response times will increase, and it is likely that technology can rise to the challenge through expanded use of electronic switching and storage technologies. Overall system average I/O response time will fall at an estimated rate of 13% per year over the next several years. However, the wide variations observed in the 1985 Customer Study are likely to continue. Not every application requires the same response time and, while low response time never hurts, it may not be free.

2.4 Summary of the Analysis of Change

This completes the analysis of parameters; their historical trends and forecast course are summarized in Table 1.

Table 1. Summary of Workload Changes

	ACC/SEC MIP	GB MIP	RESPONSE TIME
'85 BASE	15	3.7	30
'80 to '85 (History)	-10%	0%	-9%
'85 to '95 (Forecast)	-7%	2%	-13%

All percentages represent annual rates of change.

Now it is only a math exercise to convert this table into what an the average customer from the 1985 study would look like over time. For this exercise, we'll arbitrarily assume that an "average installation" grows its MIPS at a 30% annual rate.

Table 2. An Average Installation Over Time

	MIPS	GB	ACC/SEC	RESP TIME	ACC/SEC per GB
1985	35	130	525	30 ms	4.1
1990	130	510	1387	15 ms	2.7
1995	483	2092	3708	7.4 ms	1.8

3.0 TWO CLASSES of DATA

While Table 2 presents a good picture of average rate and response times across an installation, it gives no hint of the wide variations that were apparent in the response times of Figure 4. The Customer Study, with detailed access rates and response times by pool, provides some interesting insights. There appear to be two general types of data in these installations. The busiest type is "system data" comprised of the key system, spool, and scratch data pools; less busy is the "user data", a combination of primary application, secondary application, batch, and off-line data pools. With some installation

variation, the system data averaged close to half the accesses while occupying only 14% of the capacity. It had an access density of 13.3 accesses per second per GB compared to the user data at 2.6 accesses per second per GB (see Table 3, column 4, 1985 heading). With this disparity in access density, a factor of more than 5, it is necessary to work with the components rather than try to draw conclusions from to overall averages. The distinction between system and user data will be made in the upcoming discussion that matches devices to workloads.

Examining the Customer Study data reveals a weak trend that suggests larger systems have slightly less system data proportionately than smaller systems. Over time the exploitation of large memories should lead to the increased capture of very active system data. For example, ten years ago much of the task, resource, and network management information now commonly held in processor memory was shuffled back and forth to DASD. In this paper, system data is projected to grow only 75% as rapidly as user. Table 3 depicts the scenario.

Table 3. Details of an Average Installation Over Time

1985 35 MIPS	GB	ACC SEC	RESP TIME	ACC/SEC GB
SYSTEM DATA	18	242	34	13.3
USER DATA	111	284	26	2.6
	130	525	30	4.1
1990 130 MIPS	**GB**	**ACC SEC**	**RESP TIME**	**ACC/SEC GB**
SYSTEM DATA	53	638	9	12.1
USER DATA	457	749	20	1.6
	510	1387	15	2.7
1995 483 MIPS	**GB**	**ACC SEC**	**RESP TIME**	**ACC/SEC GB**
SYSTEM DATA	158	1706	4.9	10.8
USER DATA	1934	2002	9.3	1.0
	2092	3708	7.4	1.8

The ACC/SEC apportionments between system and user data, allocated according to the prior discussion, tend to more user and less system access over time. The allocation of response times between system and user data in '90 and '95 was determined by assuming that the system data will be mostly cached in '90 and 100% cached in '95. The Customer Study captured only limited usage of cache for system data and correspondingly several of the highest pool response times in Figure 4 are for key system data. Caching studies have consistently shown that key system data is a prime candidate for cache. The assumption is that all key system devices will be attached through cache controllers by 1990. Since the bottom line overall response time is determined from the previous trends discussion to decline 13% annually, the user data response is uniquely determined.

It is interesting to observe that while 30% per year MIP growth is less than many large computer installations have experienced over the past ten years, it generates a processing capability of nearly 500 MIPS in 1995, awesome by today's terms. Further, even though the I/O rate growth is assumed to be slower than that of processing power (-7% per year in ACC/SEC per MIP), it grows to over 3700 accesses per second; another remarkable number when measured against today's systems. Meanwhile, we project I/O subsystem capacity will have grown to over 2000 GB; or measured in terms of single capacity 3380s, that is about 800 boxes or 3200 actuators!

This completes the workload projection portion of this paper. A discussion of devices and their attachment in view of the projected loads follows.

4.0 DEVICES IN THE 90's

The obvious question is: "How will storage devices in the 90's change if these projections come to pass?" This paper concentrates on the 1995 installation and deals with system and user requirements separately. Experience teaches that new DASD products are fully integrated in large installations roughly two years after their availability. The devices to handle the 1995 workload will need to be available for installation by 1993.

4.1 Devices for System Data

System devices in 1995 can achieve a response time of 4.9 ms. Assuming the data is cached, a good quick estimate is:

Resp Time = P(cache hit)*RT(for a hit) + P(miss)*RT(for a miss).

There are infinitely many sets of numbers that satisfy this relationship and yield 4.9; we need to narrow the selection with some knowledge of technology. Current caches are typically achieving read hit ratios of .87 to .92 on system data; writes are not cached. I assert that by the mid 90's, the technology to get cache equivalent performance for both reads and writes will exist and further that the combined hit ratios for both will be at least as good as is achieved today for reads. So for P(cache hit) pick .9 and for P(cache miss) pick .1 (1-.9). A higher hit ratio based on larger caches and improved designs could be argued for, but with some of the more frequently used system data migrating into processor memory, a conservative .9 is prudent. Today, cache hits are satisfied in about 5 ms. This includes device queueing, device and channel protocol, cache directory search and data transmission time. Improvements are attainable in each component, and by the early 90's a cache hit response time of 3 ms is a reasonable expectation. Filling these estimates into the equation for Resp Time above and solving for RT(for a miss), we get 22 ms. This is the response time required from the backing DASD device if an overall 4.9 ms response time is to be achieved. To see one way it might be reached, consider Table 4.

Table 4. Components of response time for a typical DASD access to system data, uncached in 1985 and a cache miss in the early 90's.

	1985	Early 90's
Device Queuing	10	4
Start I/O delay	1	1
Search and Protocol	3	2
Seek	5	5
Latency	8	7
RPS Delay	5	2
Transfer	2	1
	34 ms	22 ms

The 1985 seek estimate of 5 ms reflects the well known fact that seeks are not random. Most installations measure actual seek times ranging from 20 to 35% of published nominal seek times where the nominal are based on truly random seek patterns. The seek time estimate remains constant at 5 ms in the early 90's estimate. While device seek mechanisms will get faster, caching tends to eliminate proportionately more of the short and zero seek accesses, thus raising the observed seek time. Holding the seek time constant from the '85 to the early 90's assumes that technology improvements and the effect of caching balance out and no net change occurs. With other components showing moderate improvements in technology, the response times of 22 ms for cache misses, composed of 17 ms service time and 5 ms device and path queuing, will be reached.

Now the question is: "How much data can we place under the arm of one of these devices?" From the 1995 projection, system data at the typical installation was receiving 1706 accesses per second and occupying 158 GB of space (Table 3). Of the 1706 accesses, 90% are picked off by the cache and the remaining 10%, or about 170 become real physical accesses to the DASD devices. Next we ask: "How many accesses per second can an arm produce?" The naive answer is 1/(service time per access) = 1/.017 = 58+ accesses per second. This answer is wrong on three counts.

First, 1/(service time) assumes 100% device utilization, which is impossible to achieve and inconsistent with 4 ms device queuing. Experience teaches that an average device utilization of 15% is a good operating point for DASD in large MVS systems. Device skew (the phenomenon that some devices are much much busier than others) limits the average. Second, on a cache miss, the amount of data transferred is much greater than one record as the remainder of the track is transferred following the target record. For cache misses, use two times the latency as the transfer time; in this example the service time estimate becomes 30 ms. Starting with the early '90's base in Figure 4, remove queuing delays and transfer time and then add twice the latency (22-4-1-1+14). Third, handling writes in the cache implies some destaging will occur as updated tracks fall to the bottom of the LRU stack. While these destages are not seen as part of device response time they do increase device busy just as staging the remainder of the track after a miss does. Without a derivation I will claim that a good approximation to the probability of destaging is:

P(access causes a destage) = P(write)*P(miss)

This is not an exact formula, but under normal conditions is reasonable. In the current example, P(destage)= (.125)*(.1)= .0125. So device busy time of 30 ms for misses must be increased by 30*.0125 = .4 to account for destage activity. This is not within the range significance for a "back of the envelope" calculation and will be ignored. It could become significant with lower hit rates or lower read-to-write ratios.

Putting these factors together gives a more accurate but still back of the envelope estimate of access rate per arm of .15*/.030 = 5, and to service 170 requests per second requires 34 arms.

How much data per arm? 158 GB of data on 34 arms implies we can afford 4.6 GB per arm and still accommodate the average system! However, a design to just satisfy the average system is not prudent. For DASD, a margin of twice the average seems to cover almost all of the MVS installations. By this rule, a volume with 2.3 GB is indicated as reasonable for cached system data.

This allowable capacity per arm for the 1995 workload is large by current standards and sufficiently interesting to warrant further investigation. One cross check is to look at the 1985 workload in the same way and find an allowable capacity under the arm for it. With no caching and an I/O rate of 242 accesses/second spread across 18 GB of system data, an average device utilization of 15% and a service time of 23 ms, we get: .15/.023 = 6.5 accesses/second per arm. The 242 accesses/second require 37 arms and spreading 18 GB across 37 arms gives an allowable capacity of .5 GB per arm. Comparing .5 GB calculated with the 3380 single-capacity devices at .63 GB indicates that the fit is roughly correct for the average system; but uncached 3380's holding purely system data may be a little too big for some systems. This matches quite well with contemporary customer experience where many installations currently use cached single capacity devices for system data. A design point at half the requirement (i.e., .25 GB per arm) would not be a cost effective solution. The cost of such a device would be much greater than half a current single-capacity device and cache is quite effective for those systems in the upper half of the requirement range.

A note of CAUTION: The methodology above is only an approximation. It is not a valid capacity planning methodology but a "ballpark" estimate of what type of storage devices might be appropriate to match against a broad class of demand.

4.2 Devices for User Data

While system devices were projected to have an average response time of 4.9 ms, user devices are projected to reach 9.3 ms. First, some caching is certainly required to achieve this objective, since the 1995 projection for cache misses on system data is 22 ms. User data having lower access rates may experience less device queueing. However, since we are looking to configure large capacity devices and run them at comparable utilizations to the devices

holding system data, the queueing delays will be comparable. To get an idea of how much caching is required, suppose that user data, being somewhat more unpredictable than system data, experiences a cache hit probability of .87. Then if we use an extension of the analysis for system data we get:

```
User data RT = P(cached access) * RT(for cached
               data) + P(uncached access) *
               RT(for uncached data)

   RT(for cached data)=P(hit)*RT(for a hit) +
                       P(miss)*RT(miss)
                      = .87  * 3 ms   +
                        .13  * 22 = 5.5 ms
```

RT(for uncached data) is the same as response time for a cache miss except that a slightly lower seek time is expected per the prior discussion. Using a seek time of 4 ms rather than 5 ms in the cache miss case in Table 4 gives 21 ms total. So

```
User Data RT = P(cache)*5.5 + (1-P(cache))*21
             = 9.3
```

This gives P(cache) = .75. It means that 75% of the accesses to user data have to be to cached devices, but not necessarily that 75% of the devices will be cached.

Since the expectation is that wide variations in response time will exist from installation to installation, it is interesting to look at both the cached and uncached devices in terms of arms required and how much data can comfortably go under an arm. Consider cached devices first.

From the system data analysis, an arm can handle about 5 cache misses per second. With a hit ratio of .87, that gives a total rate of 5/.13 = 38 accesses per second to the logical device. The 1995 table indicates that user data experiences 2002 accesses/second, so 2002/38 = 52+ user devices are required for thruput. If the projected 1934 user GB are spread over 53 devices, that would yield an amazing 36.5 GB per access mechanism for the average system. The half size rule to cover most systems yields a capacity of 18 GB. In all likelihood devices will not be nearly that large. If they were, seek and latency might have to be much bigger as the physical dimensions of such a device would likely dwarf today's HDA's (Head-Disk-Assemblies). Optical storage might produce such capacities, but current technologies give little promise of meeting the 17 ms (22ms RT - 5 ms device queueing and SIO delay) service time projected for magnetic DASD and required to make this projection methodology valid.

For uncached devices, things are more as we expect. At 15% utilization and with a service time of 16 ms, we get .15/.016 = 9.4 accesses/second per device as compared with 36 for cached devices. The projected 2002 access rate requires about 213 arms. Spreading 1934 GB across 213 arms yields 9.1 GB per arm for the average system, and the half size design rule gives 4.5 GB per device. That is almost four times the data per actuator of double capacity 3380's.

Another sanity check against the 1985 data is in order. Table 3 shows 1985 user data at 111 GB with 284 access/second. Uncached devices in 1985 have a service time of 23 ms and can handle .15/.023 = 6.5 access/second. Thus, 44 devices are required (284/6.5); and with a capacity of 111 GB this translates to 2.5 GB per arm. This is roughly two times the double-capacity 3380E and corresponds exactly to the rule of thumb for a DASD design to cover the broad range of requirements by having about half the capacity that would just satisfy the average installation.

Remember, the 1985 and 1995 workloads are for average large system MVS configurations during daily peak time periods. Many installations have above average workloads or average loads with above average peak stress periods. Good design dictates that a somewhat over qualified product is far more desirable than a marginal one. One further caution, VM systems appear to generate higher ACC/SEC per MIP workloads. While the general methods presented in this paper are applicable for VM, the specific values are not.

5.0 CONCLUSIONS

This paper forecasts that I/O subsystem workloads in large MVS installations will continue to grow over the next several years. While growth in capacity will exceed growth in MIPS, growth in access rates will not keep up with the growth in MIPS. I/O response times will continue to improve. To accommodate these trends, DASD devices and their attachments will evolve. The primary effects will be movement to cached subsystems and larger capacity devices. This is summarized in Table 5.

Table 5. Evolution of DASD Workloads and Devices

WORKLOAD	1985 to 1995
ACC/SEC per MIP	52% REDUCTION
GB/MIP	22% INCREASE
RESPONSE TIME	75% REDUCTION
DEVICES	**1993**
GB/ACTUATOR	
System Data	2.3 GB (Cached)
Users Data	Up to 18 GB (Cached)
	4.5 GB (Uncached)

6.0 ACKNOWLEDGEMENT

This paper draws heavily on a customer study done by Marti Henley and several people in her department including Ingrid Lue, Bruce McNutt and Kendall Wright. Without that study, this paper could not have been written.

7.0 CITED REFERENCE

1. J. B. Major, "Processor, I/O path and DASD configuration capacity," IBM Systems Journal, Vol 20, No. 1, 63-85 (1981).

CACHE MANAGEMENT IN A HOSTILE ENVIRONMENT

Rob Wells
National Advanced Systems

ABSTRACT

A discussion of the problems facing management to reduce response times in an unfriendly software environment and the strategies adopted to overcome these problems is presented. The implementation of a cache DASD subsystem is an integral part of this plan.

The significant benefits obtained were predicted with the use of empirical methods. A discussion of the additional tools required to implement cache support in such a hostile environment will also be presented.

1.0 INTRODUCTION

All data processing organizations experience performance problems at some point or another, but what alternatives are available for those users that are behind in their implementation of software technology? Examples of this are users continuing to use outdated or unsupported operating systems, like VS/1 or DOS/VS.

This paper discusses a case of a VS/1 user maintaining continuity with hardware advances that are reportedly not supported. First is a discussion of the problems facing management in a "hostile" user software environment, and an approach that was undertaken to alleviate it. As part of the approach, a case is built for the implementation of a cache controller. Second, a discussion is presented describing the implementation of a tool to support cache in a hostile unsupporting VS/1 environment. During each of these discussions, a description of the operation of a cache controller is presented.

2.0 PROBLEM DEFINITION

2.1 SYMPTOMS

As in most installations, when the phone begins to ring, it is usually the first indication that there is some type of problem. In the computer industry, most problems relate to either service or performance. Problems relating to performance are broken into two categories, perceived and real.

Perceived problems are usually caused by a lack of service level agreements with the users. Without them, user satisfaction can vary greatly, based on almost anything, including what day it is. On the other hand, real problems are caused when the service level is not met, whether the agreement with the users is implied, verbal, or written.

In the case of the installation from which the study is drawn, the problem is real, and the service level is implied. Dissatisfied users were experiencing an online terminal response time of between 15 and 30 seconds.

2.2 PROBABLE CAUSE

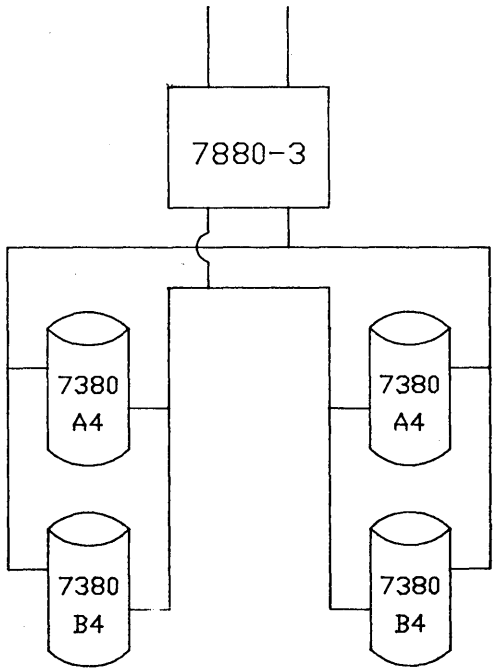
Discussion of the processing profile of a typical online inquiry transaction revealed that the I/O processing involved a complex search of a data base index, and this was responsible for the major proportion of the transaction response time within the CPU. Confirmation of this process being a major bottleneck causing excessively long and variable response times was undertaken using basic queuing theory. The queuing analysis revealed that a mere 10 percent reduction in the total I/O response time for the index search was likely to reduce the transaction response time to an acceptable and stable 4 seconds.

2.3 THE ENVIRONMENT

Before considering possible solutions to the problem, it was necessary to understand the environment in which it arose. The environment in this case was made up of three sections, the software environment, the hardware configuration, and characteristics of the index search.

2.3.1 **The Software Environment.** The software environment was potentially a problem in itself. Knowledge of the intimate details of the operating system, VS/1, and the software applications was possessed by very few. The software was already highly modified for local purposes and poorly documented. This situation could be seen as a hostile internal environment in that there were limited resources to resolve a difficult problem.

2.3.2 **The Hardware Configuration.**



2.3.3 **The Characteristics of the Index Search are:**

- Device type	7380
- I/O's per search	multiple
- Access Method	EXCP
- CCW's	Chained
- Blocksize	3520
- Blocks/EXCP	12
- Threading	Single
- Granularity of lock	Whole Index

3.0 POSSIBLE SOLUTIONS

It was necessary to develop a list of possible alternative solutions to the problem and then determine the impact of each. Due to the technical nature of the problem, the list was comprised of possible technical solutions. Without explaining each in detail, they are all suggestions made by the technical staff that concern improvement to the search of the data base index. There were:

-split the index across multiple volumes

-provide separate index I/O path

-reduce lock granularity

-introduce multiple threading

-make index (parts) resident in real CPU memory

-introduce cache controller

4.0 EVALUATING THE ALTERNATIVES

4.1 MANAGEMENT ISSUES

Once the problem was confirmed to be real and its likely source isolated, management had to begin to take steps to correct the problem in both the short and the long term. The next step in the process was to identify the management issues involved in its resolution. In this case, they were rather simple:

1. **User Satisfaction**. Users were frustrated and potentially hostile to the extent that a slow terminal response time was occurring. Only crucial work was being performed and productivity was low.

2. **Service Level Improvement**. To enhance the environment it was necessary for the service level to improve to stimulate productivity.

3. **Service Level Maintenance Once Improved**. Maintaining the service level was important for planning purposes. It was expected that in addition to normal growth, there would be a high degree of latent demand due to a potential increase in the productivity level. The thought behind this was, enhancing user productivity would stimulate use of the system.

4. **Resources Required to Resolve the Problem**. This consideration was felt to be significant because in this installation the quantity of technical staff was limited. Alternatives such as consultants could be considered to complement the lack of staff, but time, cost, and availability were factors especially due to the existence of an outdated VS/1 based system.

5. **Cost Effective Solution**. Cost in any environment is a consideration. Cost here is made up of three important entities: time, materials, cash, and staff resources. The need for each would be weighed accordingly to their respective usage in the possible solutions. Cash resources were available provided total costs could be identified in advance.

4.2 TECHNICAL ISSUES

4.2.1 **Understanding Cache Concepts**. Before much is said regarding a case for cache, a brief description of what cache is and its operational characteristics is required. A cache is defined in general terms as a hiding place, typically for treasures or provisions. As it is used in the computer industry, it is a high-speed buffer for the transfer of data from a slow medium to a faster one. In its original use, cache acted as a buffer between the CPU and Real Memory to improve the performance and throughput of the system. With the reduction in cost in producing high-speed memory, cache has been implemented in the DASD I/O subsystem to benefit its performance.

Cache in the DASD I/O subsystem resides in the controller between the high-speed channels and slow, rotating disk devices. Cache's performance benefit is based on the assumption that a sufficient portion of the data being retrieved resides in the cache buffer. Given a high proportion of data being found in cache, the effect on performance can be overwhelming. The DASD I/O response time for a typical 3380 subsystem can be reduced from around 25 milliseconds to 3 milliseconds, if the record being read resides in cache. The average response time can be halved. Based on a reduced average response time it is possible to achieve higher channel utilization in cached DASD subsystems than in non-cached subsystems, while retaining acceptable response times.

There are four basic operations that can occur in a cache environment. They are:

Read Hit

This is the most important of the cache operations. A read hit occurs when the record of data being read exists in cache, and therefore can be passed immediately to the channel, resulting in a minimum response time for an I/O operation. In this case, no access to a disk is required.

Read Miss

For this operation, the record being sought is not in cache; therefore, a read operation from a disk volume occurs. In addition to the record being read, stored in cache, and simultaneously transferred to the CPU via a channel, a prolonged operation transfers either the rest of the track or the entire track into cache, and makes all of that data available for possible READ HIT retrieval later. The reason for doing this is that the probability of requiring another record from the same track has been observed to be high; therefore, the next read operation is likely to result in a read hit operation and the overall average response time will be reduced.

Write Hit

For a write hit operation the record being written exists in cache and, therefore, is updated in cache and on the disk volume simultaneously. For data integrity reasons, the proper condition code of the write operation is not issued until the write operation to disk is completed. If the record is reread immediately, then the record is found and retrieved from cache in a READ HIT operation.

Write Miss

In a write miss operation, the record does not exist in cache. Cache is completely bypassed and the record is written to disk only.

4.2.2 **Software Support of Cache**. The above functions occur transparently to user programs, and the operating system. For this reason, cache operations can be implemented in environments that are normally thought of as not supported, such as VM non-HPO, DOS, or VS/1, provided that caching of the volumes of interest or in need of cache support can be achieved without the usual operating system software. In practice, this proves to be possible.

It is important to note that the default situation in a cached environment is that all volumes behind the cache controller are cached. Whole tracks containing records read are moved into cache as described in READ MISS, and subsequently the benefits of READ HIT can be obtained for associated volumes. As a consequence, the selection of volumes to reside on the cache string must be more careful than usual as the achieved Read to Write Ratio has a significant impact on performance, as does the Read Hit Ratio. If acceptable levels of the two ratios are not maintained, performance can be adversely impacted.

Given that VS/1 does not support a cache DASD controller, this is another reason to define the environment as hostile.

5.0 THE CHOSEN SOLUTION

When making any decision, the alternatives must be weighed against the issues that have been defined, and then one chosen that satisfies most or all of the requirements with the least impact on resources.

It is not the purpose of this paper to report on the evaluation of each of the alternative solutions considered. Several of them have been implemented over a period of time. Rather, the solution which was considered likely to be the fastest and easiest to implement, and therefore to be attempted first, is discussed in detail.

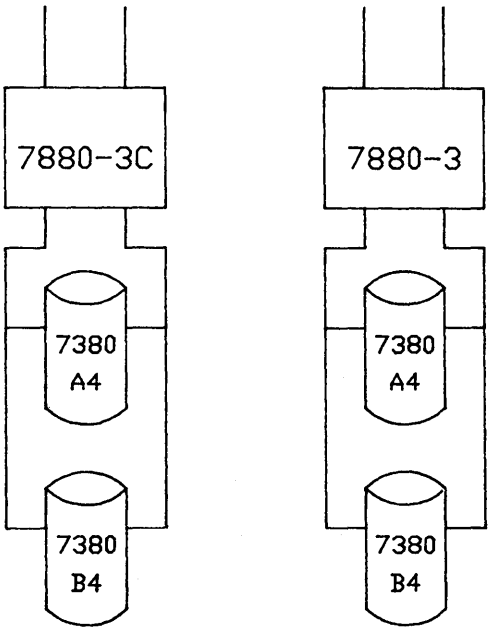
The introduction of the cache controller was accepted as the first alternative due to the potentially beneficial impact on performance, minimal staff involvement required to implement it, and the speed with which it would be achieved. The controller could be delivered quickly and an experiment could be developed and conducted to prove its effectiveness.

6.0 TESTING AND EVALUATING THE INITIAL SOLUTION

Given an understanding of cache and its operation, it is easier to understand the following case that is presented as a successful implementation in the rather hostile environment; hostile in the sense that the selectable functions of cache controllers are not supported by VS/1.

6.1 CONFIGURATION CHANGE

The diagram that follows shows the change to the configuration with the introduction of the cache controller. With this addition, there are the effects on performance of two changes. First, there are extra paths to the current devices, and second, there is the existence of the cache itself.



6.2 EXPERIMENTAL DESIGN

The dramatic improvement of response time is usually a simple thing to measure. First, all the user complaints stop and so does the ringing of the phone, but that is not a good justification for purchasing equipment. To keep things simple and to satisfy manpower constraints, it was agreed that a full scale online test was not necessary to measure the effect of a change. A simple design was agreed upon to measure the effects of implementing a cache controller. A batch job was selected that performed a partial restore of the data base, because it closely replicated the profile of the online system's data base index activity. Performance of the job running stand alone would be measured on elapsed time only. The percentage of improvement between different executions using different DASD I/O configurations was expected to give a general indication of relative performance for those configurations in the online environment.

First, a Base Line of performance was determined to establish whether improvement or degradation occurs. Second, a method was established to measure the effects of the changes. In this case, three change effects were measured:

1. **Base Line.** Both the index and data base were located on volumes on one DASD string headed by a non-cache controller. This was essentially equivalent to the pre-existing configuration described in section 2.3.2.

2. **Configuration Effect.** The addition to the DASD I/O subsystem of another controller in itself has an effect, so this has to be measured. The configuration effect test placed the index on one string, and the data base on the other string. The cache was disabled by hardware means, and (index) I/O performance difference was reflected in the changed job elapsed time. This provided a measure of the configuration effect alone.

3. **Cache Effect.** To enable the effect of cache on all the data being accessed, both the index and data base were placed on the same string connected to the cache controller. Here any change in elapsed time for the job was attributed to the existence of the cache alone.

4. **Combined Effect.** To measure the combined effect of the configuration change (second DASD controller) and the introduction of cache as a presumed improvement to the index search I/O processing performance, the index was placed on the string with the cache, and the data base was placed on the string with the normal controller. This was done knowing that the index was a file that was being read significantly more than it was being written, with several records being read in sequence.

6.3 RESULTS OF THE EXPERIMENT

TEST EFFECT	ELAPSED TIME	PERCENT IMPROVEMENT
Base	29:34	--
Config	29:05	2
Cache	23:36	20
Both	21:51	26

6.4 EFFECTS ON THE ONLINE SYSTEM

After completing the experiment, the cache controller was left installed for use in the online environment. The online response time was reduced from the 15-30 seconds at peak times to approximately 4 seconds during these peak periods, a minimum of a 75 percent reduction.

As predicted through the queuing theory analysis, just a slight improvement to the service in the data base index queuing had a large impact to the overall response time.

6.5 EVALUATION OF THE INITIAL SOLUTION

The introduction of the cache controller proved to be very effective. It satisfied the users, provided a significant service level improvement, and staff resources required to implement the solution were minimal. The costs involved were effectively limited to those involved in purchasing and installing the cache controller. Consequently, the implementation of the solution chosen initially, satisfied the dual constraints of minimal use of site staff and limiting the cost to a predicted level.

Additionally, an unexpected benefit was realized with this implementation. With the bottleneck in the I/O subsystem now alleviated, the CPU Utilization dropped slightly. This permitted the delay of a CPU upgrade for several months, effectively buying back CPU capacity, also offsetting the cost of the cache controller.

7.0 FOLLOW-UP

7.1 OUTSTANDING ISSUE

The alleviation of the DASD I/O bottleneck may have satisfied and calmed the users, and also stopped the phones from ringing, but one major management issue still needed attention. This issue is maintaining the service level once it is improved. In a stable environment, and with the removal of the bottleneck, one could expect that the problem would not resurface. In this installation, however, the business growth rate is approximately 15 percent per year. With this in mind, it can be expected that the problem would return, either in the same form or perhaps as a new and different bottleneck, as the demands for access to the data base increase.

There were two parts to this problem. First, there was a need for periodic monitoring of the performance of cache to ensure that correct data with the right usage characteristics was being cached. Second, a method was needed to control or select the devices under the management of the cache.

To accomplish this, a method to provide directions to the new controller had to be developed, since VS/1 does not support cache commands. In fact, VS/1 prevents the issuing of cache commands. There were several problems that native VS/1 caused, which brought back the issue of the hostile software environment. The major problem that was discovered was that the operating system was validating the CCW commands prior to executing them. This meant it was not possible to issue commands either to select volumes for caching individually or to read out usage statistics maintained by the controller hardware and microcode.

7.2 CACHE OPERATION

Prior to describing the approach to these issues, a further discussion of the operation of cache is required. In addition to the types of I/O operation discussed earlier, there are three modes in which a cache controller can function. The modes as described below are selectable at the disk unit level and are used to control the performance of the device in accordance with the type of data on the volume and the way in which it is being used. They are:

Basic Mode

In basic mode, all records are considered to be randomly accessed, although the probability of accessing two or more records from the same track successively is considered high. Records and the remainder of the track where they reside are read into cache, replacing data in cache on a Least Recently Used basis.

Sequential Mode

When records are being read in sequence and a record is brought into cache, the controller pre-stages additional records by also reading into cache up to four tracks, depending on the manufacturer. With continued pre-staging, data can then be transferred more quickly from the controller without waiting on the spinning device.

Inhibit Mode

Under inhibit mode, cache operations are completely bypassed. This mode is selectable at the device level or on a whole subsystem basis.

7.3 CACHE SUPPORT

When an operating system supports cache, software is used to control the functions described above. The supporting environments are MVS/SP or MVS/XA with IDCAMS, and VM/SP HPO and SF. The functions that are performed by software interfaces can be broken into two categories:

1. Those functions that enable or disable the use of cache when accessing data at the disk volume, or entire subsystem level. These functions offer the capability to optimize the cache access environment.

2. Those function which collect statistics regarding the number of accesses to cache in each mode of operation, and also collect byte counts for every access type. This information is transferred to the system in response to special channel commands and is useful in developing operating and performance ratios of cache for each enable volume.

These functions are implemented via special channel commands. There are five primary commands to control the new functions:

1. **Define Extent**. Specifies access range and access modes for cache. The command code is '63' and it transmits the access area and mode.

2. **Set Subsystem Mode**. Determines availability of cache memory for pertinent devices or the whole subsystem. The command code is '87' and transmits enable or disable information.

3. **Sense Subsystem Status**. Logs cache status information indicating the operating status of cache. The command code is '54' and the controller transmits 40 bytes of information.

4. **Sense Subsystem Counts**. Logs input/output operations counter data indicating the operating status of cache. The command code is '74' and the controller transmits 80 bytes of information. This information is discussed later.

5. **Set High Performance Storage Limits**. Bind all records in a specified area from a nominated track in cache or release that binding. This function is only implemented on the model 13 controller. The command code is '3B'.

7.4 CACHE STATISTICS

The statistics that are maintained and retrievable provide valuable information for the tuning of the cache subsystem. The types of data that are collected and retrieved via the Sense Subsystem Counts channel command are:

For Both BASIC & Sequential MODE

-Read I/O count

-Read Hit I/O counts

-Write I/O counts

-Write Hit I/O counts

Other Counts

-Load Inhibit I/O counts

-Read I/O on bound tracks

-Write I/O on bound tracks

-Cache load count

-Byte count from Read Hit

-Byte count from Read Miss

-Byte count from Write Hit

-Byte count from volume to cache

These statistics can be used in the development of Read to Write ratios, READ HIT ratios, and monitor the load on and performance of the cache. Without describing each statistic in detail, it is important to know that these values are available and important in maintaining optimum throughput in the cache subsystem.

7.5 GOOD CACHE CANDIDATES

With an understanding of the ability to control the operation of cache, it is now important to understand the characteristics of those volumes which make good cache candidates. In general, the following characteristics make for excellent eligible devices:

1. The number of READs is high compared to WRITEs, where the device is reading greater than 65-70 percent of the time.

2. The frequency of access is high, approximately 2000 I/O's per hour or greater.

3. The tendency of completely random access is not very high, proximity of reference is close.

7.6 OTHER BENEFITS OF CACHE

With the capability of inhibiting the use of cache on unfavorable volumes, installations can maximize I/O throughput given proper management of the I/O subsystem. In addition, the cost of an effective megabyte of storage can be reduced by enhancing the following tendencies:

1. Increase the number of devices on a string to the maximum.

2. Enable the use of double capacity devices.

3. Increase both the allocated and effective usable space on each volume.

In the absence of cache controllers, installations usually limit the number of devices on a string to eight or twelve, and the number of controllers per channel to one, for performance reasons. In addition, installations use a general rule of having a maximum of 80 percent of the space on a volume allocated in an effort to provide for performance benefits.

With the introduction of cache, these limitations can be minimized or eliminated, thus recovering data space on devices, for example, and reducing the overall cost of storage.

8.0 THE FINAL APPROACH

8.1 UPGRADE THE SOFTWARE

The installation was also looking to install VM/SP to support a development virtual machine and an 'E' model double capacity disk. The need to issue cache commands was another reason to do so. Although VM/SP non-HPO also does not support cache commands, it introduced a means to get around CCW validation. Non-HPO VM/SP was selected because of its low cost relative to the HPO version.

A program was created to parse input from a console and issue the new channel commands to the controller. The intention was to closely simulate the function that IDCAMS performs when it controls cache under MVS/SP and VM/SP HPO. The program development was not as timely as the implementation of the cache controller itself; however, costs were minimal.

Immediately the two problems were solved, controlling and managing the devices on the string, and collecting statistics on the operation of the cache.

9.0 SUMMARY

Management succeeded in addressing all of the issues that were originally defined. The dissatisfaction of the users subsided with the reduction of the terminal response time. The service level improved to a point that it facilitated enhanced productivity and was acceptable. Maintenance of the service level was enhanced by the development of a program to support the solution, providing the capability to plan and monitor demand. Staff resources required in the implementation were minimal, and finally the costs involved were held to a predictable and reasonable level.

Overall, the implementation was highly successful and provides a model for other installations in the same hostile software or hostile user environment. The experience was informative in understanding the operation of cache, and how advances in technology can continue to lend solutions to better business and increase productivity.

10.0 ACKNOWLEDGEMENTS

Special thanks are due to Jeff Perkins who assisted greatly in the development and implementation of the cache program under VM. Thanks are offered also to all the technical staff who worked hard to conduct the experiment, and to John Temperly who assisted greatly in the editing of this paper.

11.0 REFERENCES

1. IBM Publication GA32-0067 "IBM 3880 Storage Control Model 13 Description".

2. IBM Publication GA32-0083 "IBM 3880 Storage Control Model 23 Description".

LASER OPTICAL DOCUMENT MANAGEMENT SYSTEMS AND THEIR CAPACITY PLANNING IMPLICATIONS

David E. Y. Sarna
International Systems Services Company
New York, NY

ABSTRACT

Laser optical imaging has emerged from the laboratory and is being productively used in a growing number of commercial sites. Scanning of images and their subsequent storage and retrieval from laser optical disk subsystems presents opportunities for revolutionizing document processing. At the same time, it poses challenging problems in capacity management, requiring new metrics and instrumentation, while also drawing on the profession's accumulated experience in hierarchical storage management.

This paper will present an introduction to these devices, their performance considerations, and related cost metrics, and will report on our experience with several pilot installations.

INTRODUCTION

A subscriber calls an insurance company. He asks the status of a claim. The customer service representative asks the subscriber for a name, and date of birth. Within seconds the service representative has the entire file available, and tells the subscriber that his claim was received yesterday, but that he did not sign his claim form. A facsimile of his form was mailed out yesterday evening for him to sign and return.

Too expensive? Impractical? A revolution going on right now in document processing is expected by many to have a more profound effect on American business than even the personal computer revolution which took place at the beginning of this decade.

The mail arrives. Automatic equipment opens it, and removes and batches tax return documents. Documents of all shapes and sizes are carried by a moving belt. As they reach a scanning head, forced air holds them flat, and they are scanned on both sides. Some of the returns have been typed or prepared by computer. The image of these forms will be sent automatically to intelligent optical character recognition equipment, which will read and extract the text where possible. The returns and supporting documentation data will be compressed and then stored on an optical laser disk. After completing any additional data entry, an expert system will review the documents' extracted text, and if there are any questions it will send the file electronically to a tax examiner. If a return is incomplete or rejected, a letter will be generated to the taxpayer together with a facsimile of his return with the incorrect or missing data clearly highlighted on the facsimile. It will be possible for authorized personnel to review the entire file, without ever touching or routing paper.

The technologies that make these scenarios possible and cost effective are:

- High-speed optical scanners,
- Intelligent optical character recognition equipment,
- Laser optical disk storage and juke box device handlers,
- Image compression hardware/software,
- Laser printers,
- High-resolution display stations,
- Fast PC-based workstations, and
- Expert systems software.

What is another example of a potential use for this technology?

An ambulance driver wheels a patient into an emergency room. An orderly keys in a name and date of birth. Within seconds, an electronic version of the patient's file, showing his two previous admissions, is available. The patient's allergy to a particular drug is immediately noted, and his vital signs can be automatically compared to his past history; charts, electrocardiograms, etc. can be added to the electronic file throughout the patient's stay.

How can all of the hospital's patient files be economically kept for retrieval quickly without prohibitive cost? A single removable laser optical disk has space for about 50,000 documents. But a single juke box can hold several hundred disks "near on-line". Like the old fashioned juke boxes, a "near on-line" pack can be picked and mounted in a matter of seconds. Any of ten million documents can be retrieved quickly from a single jukebox, and multiple jukeboxes may be controlled by a single system.

The foregoing are but a few of thousands of potential applications of the advanced usage of intelligent workstations. There are five key features of this type of application which represent a major advance over earlier optical disk-based archival storage and retrieval systems:

- Full *integration* of the process with conventional mainframe-based application systems,
- Use of embedded *expert systems* techniques, where appropriate,
- Use of integrated *optical character recognition* techniques, as appropriate,
- Processing based around *industry-standard hardware and software*, and
- Tailored *application systems* capabilities based on in-depth industry expertise.

Technical factors have emerged, which – in combination – have the potential for dramatically increasing the capabilities of intelligent workstations. Among these are:

- Mini/micro computer price performance has improved dramatically,
- Optical scanners have improved in both speed and accuracy,
- Optical laser disk storage has become practical for mass storage,

- Local Area Networks (LANs) have improved minicomputer, microcomputer, and mainframe computer inter- and intra-connectivity, and
- Expert systems techniques have found their way into practical business applications.

An illustrative example of the integration of these technologies in an advanced scanning, editing, and computer-assisted retrieval application is shown as Exhibit 1.

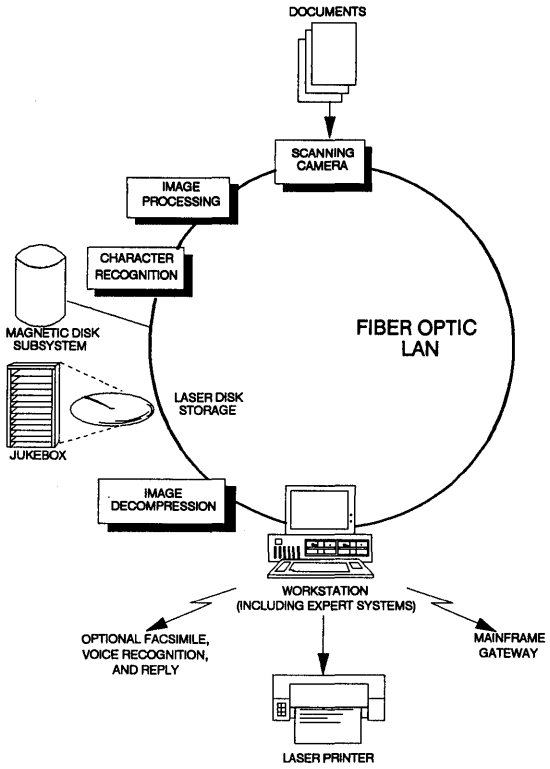


Exhibit 1. Document Management Systems.

Many believe that laser optical technology will, by 1990, have a more profound effect on society than even the advent of the personal computer in the 1980s, and according to Rothschild Consultants, will result in a $5.8 billion market.

Laser optical systems,when fully integrated with information processing, hold promise for finally eliminating the duality which has plagued the information processing industry since its inception. Everywhere we find high-speed electronic systems for retrieving and processing a subset of an organization's information needs. In parallel, each organization maintains a totally separate and manual system for processing, filing, and retrieving paper-based documents. Although computer-aided microform retrieval systems have been on the market for a number of years, they have never been fully integrated and have had limited impact. Enter Document Management Systems. Advances in the technologies discussed above have, in combination, made possible cost-effective and technologically feasible automated Document Systems. Feasibility analysis of these systems has begun in many organizations. In particular, traditionally large users of microfilm (e.g., in the financial, insurance, and banking industries as well as in the public sector) have been experimenting with this technology in pilot programs for the last several years. Within the next twelve months we can expect to see several large-scale implementations come on-line, with additional systems being implemented at an accelerated pace over the next several years.

Automated Document Management Systems require an understanding of several new technologies, and present a number of novel capacity planning problems. This paper provides an overview of each technology from the points of view of the capacity planner, performance engineer, and systems architect, and suggests key issues which must be addressed in order to build a successful system.

SYSTEMS OVERVIEW

The typical system shown in Exhibit 1 shows all of the technologies in concert. From the end-user's point of view, the key difference between this approach and the classical terminal is apparent in the delivery vehicle, which is usually a PC-based workstation connected to a high-resolution screen suitable for displaying documents. Workstations are linked via an optical local area network. Files are stored on laser optical disks contained in automated juke boxes. Documents typically enter the system directly from the mail-room, where images are captured on high-speed optical scanners. Intelligent optical character recognition processors can be used to extract text from the scanned images. Expert systems are in use throughout the process to reduce the need for manual or special processing, and to increase accuracy. Laser printers provide facsimiles of images wherever a "hard copy" is required. Each of these technologies is discussed in more detail in the sections which follow.

SCANNERS

Documents typically enter the system by being scanned, through a process known as "image lift." A wide variety of scanners are available ranging from desktop systems costing under a thousand dollars, to high-speed, heavy duty units costing upwards of seventy-five thousand dollars. In selecting a unit, there are several issues to keep in mind:

- resolution,
- compression,
- grey scale,
- slew rate,
- feed mechanism,
- face and size of image lift,
- duty cycle, and
- overall capacity.

Resolution

A scanner basically transforms images to bit patterns. In general, the greater the number of dots used to represent an image, the more faithful the resolution. Typically, resolution is expressed in dots per inch (dpi), and typical scanning resolutions are 400, 300, 150, 100, and 75 dpi. Most scanners are capable of scanning at multiple resolutions, under program control. Quality typesetting is usually accomplished at a resolution of at least 1,200 dpi; today's laser printers generally have a maximum resolution of 300 dpi, with 600 dpi units just emerging. Why not always scan at the highest resolution possible? The answer lies in the size of the bit-mapped image which is generated. An 8-1/2 X 11 inch document has 93.5 square inches. A resolution of 400 dpi, therefore, requires 160,000 bits per square inch, or nearly 15 million bits (nearly 2 megabytes) per page. On the other hand, at 75 dpi the same document requires only 64 Kbytes. While the tradeoff of resolution and memory must obviously be evaluated for each application, typical systems capture images at 150-300 dpi.

Compression

Most scanners format the images in accordance with the standards for facsimile transmission, with the most prevalent being the CCITT standards for group 3 and group 4 transmission. These standards provide for a means of bit compression, taking advantage of the fact that if much of a document is white space (at least 70% in most instances) then rather than recording each picture element as a binary zero if white and a binary one if black, it is possible to record only state changes from black to white, and the distance between one state and the next. Such compression techniques, particularly for higher scanning resolutions, can frequently achieve reductions of between 10:1 and 20:1, depending on the amount of white space on the page, and the compression technique used. The group 4 algorithm, while usually more effective, is substantially more complex.

Hardware compression boards for PCs are available, based on the AMD chip, but they are still expensive, costing about $1,500. These perform compression and decompression in about one second or less. Software compression algorithms which we have tested have varied in speed from 28 seconds or more to as little as three seconds on an 8 MHz IBM PC-AT, based on the effectiveness of the implementation technique.

Grey Scale

Line drawings and text are represented as either black or white, while pictures obviously need to be presented in shades of grey. Using a process known as half-toning, which is the method used to present pictures in most forms of printing, several scanners are available which can preserve a grey scale, using either 16 or 64 shades of grey. Of course, the greater the number of shades of grey the greater the number of transitions between black and white, and the less the image can be compressed.

Slew Rate

In some scanners, the document is held in a fixed position while the scanning head moves over the paper, a line at a time. In others, the scanning head is stationary and a transport mechanism carries the document. The rate of travel is known as the slew rate and is typically measured in inches per second. It is one element in the calculation of throughput per hour, and is typically in the range of 1-6 inches per second.

Feed Mechanism

Inexpensive scanners may be completely manually fed. More expensive units are automatically fed with a feed capacity of 20- 50 documents. High-speed, heavy duty units often have a vacuum (suction) feed, and are thus less subject to misalignment. They may also have high-capacity feeders and hoppers. The highest capacity, highest cost units use a continuous feed process.

Face and Size of Image Lift

Scanners have either single- or dual-sided image lift capabilities. Dual image lift may be considered even in applications not requiring storage of the second side, so that the problem of upside down feed is eliminated. Most scanners have a capacity of documents up to legal size; larger units can accept 11 x 17 or even larger D-sized engineering drawings.

Duty Cycle

The number of documents per month determines the duty cycle. Light duty machines may scan rather quickly (e.g., 3 seconds or less per document) but may only be rated at 20,000 documents per month, implying aggregate use of under twenty hours per month. Heavy duty units will be capable of sustained round-the-clock operation.

Overall Capacity

The capacity of a unit in documents per hour is, of course, a function of document length, slew rate, start and stop times (feed times), and unoverlapped compression time (including the time for transmission of the uncompressed image to the host processor), and may be expressed as:

$$\text{CAPACITY DOC/HR} = 3600/\left(\frac{\text{document length}}{\text{slew rate}} + \text{start and stop times} + \text{unoverlapped compression times}\right)$$

The arrival rate in documents per day divided by the number of hours effectively available for scanning divided by the scanner's capacity per hour provides the number of scanners required. To provide a reserve for peak conditions, a 60% utilization factor is typically applied. The number of units required can be given as:

No. of UNITS = (Documents per day/scanner hours per day/capacity * Utilization Factor)

In addition, backup capacity should be considered, since an entire processing operation may be shut down if documents cannot enter the system.

Scanners today are typically attached to host processors using a SCSI interface, and the documents can therefore be transmitted to the host processor at a rate of 10 megabytes per second. In most cases, the uncompressed image is sent to the host processor which then sends the image to the compression hardware, if available. In some cases, the compression hardware is built directly into the scanner.

OPTICAL CHARACTER RECOGNITION PROCESSORS

Many documents are prepared which contain printed,typewritten, or computer printed text. These can be reliably interpreted by optical character recognition (OCR) processors, which may be contained either within a scanner's covers or as a separate peripheral.

Use of OCR Processors

The OCR processor accepts as input a scanned image, and produces as output ASCII representations for characters which were extracted from the image, together with an indication of uncertainty for characters which cannot be interpreted with confidence. Many OCR processors also append a bit-mapped image of the uncertain characters, for use in a subsequent "repair" or correction process. Under optimal conditions, a valid read rate of 98-99% can be achieved, significantly reducing data entry costs. Maximum benefit can be achieved if forms and manual procedures can be redesigned so that the machine's capabilities can be optimized. For example, unwanted text on the form can be printed in optical drop-out ink. A minimum rate of 85% is considered necessary for cost-effective use of this technology.

Capacity of OCR Processors

Commercially available OCR processors can extract text with throughput speeds in the range of 200-2,000 documents per hour. Thus, since the scanner speed may well exceed that of the OCR processor, more than one OCR processor may be required per scanner. Mid-range OCR processors now cost about $20-30,000 at the low end to more than $100,000, but prices are falling rapidly.

There is no standardized interface between OCR processors and host processors, but several units use medium-speed communications protocols, e.g., Ethernet, and in these cases document transmission time between host and OCR processor may be significant, and may reduce achieved throughput by 10% or more. Calculation of throughput rates for an OCR processor is best done using controlled benchmarks which are reflective of the quality of the material to be used in the proposed application.

LASER OPTICAL DISKS

Laser optical disk drives are now being shipped in commercial quantities from a variety of manufacturers. At the present time all commercial shipments are of the non-erasable variety known as WORMs (Write Once Read Many), although eraseable laser optical disks are expected to begin shipping within the next six months. In any event, for many document management systems, the lack of erasability is no impediment, as the need is primarily for an archival or permanent record.

Currently, WORMs are available in 12 inch, 8 inch, and 5-1/4 inch formats. One vendor also offers a 14 inch size. Their primary attractions are high capacity, removability, and low cost. The 5-1/4 inch WORMs currently have a capacity of 200 MB, 8 inch media can hold up to 1 GB, and 12 inch units are commercially available at up to 2.6 GB and have been announced with capacities of 3 GB per side. The 8 inch size has a very limited following and is not expected to prevail. Significant improvements in storage density are likely to be achieved over the next several years.

Exhibit 2 shows comparative cost data, and it is clear that WORMs provide the lowest available cost for directly accessible storage.

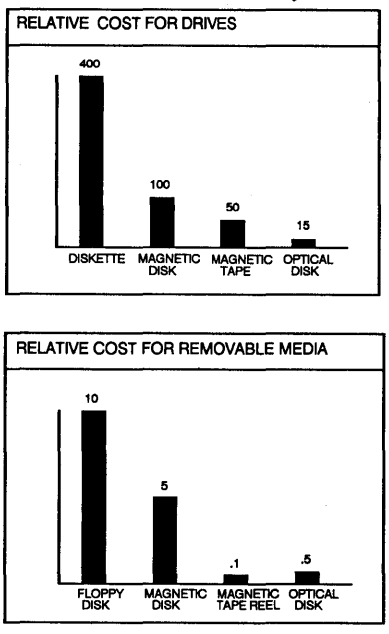


Exhibit 2.

WORMs have not yet achieved the performance of magnetic disks, owing to the greater time required to physically burn the binary representation using the laser, as well as greater settling and positioning times. Still, typical access times are in the range of .2 to 1.5 seconds. Data transfer is at this time almost universally accomplished over a SCSI interface running at 10 megabytes per second. However, more intelligent controllers, which can be directly channel-attached at speeds of 3 MB/s are reportedly on the drawing boards.

Assuming that an image is stored in compressed form at 20 Kbytes, a typical 1 GB WORM can store 50,000 non-erasable images. It is customary to use WORMs in conjunction with conventional magnetic media (disk drives) for staging and for storing indices. In some applications, file space may be preallocated, resulting in less than full utilization.

Early WORMs exhibited potential corrosion problems which led to projections of media life of under ten years; with today's reformulated compounds, manufacturers are currently quoting a 20-30 year life expectancy for data.

JUKEBOXES

Document management systems, with their relatively large record sizes (typically 20-50,000 bytes per image), can fill even 1 GB media in a short time, in high volume applications. Jukeboxes are an automatic mechanism for storing multiple platters (typically up to about 100) in a storage frame containing slots for each platter, in much the way trays are stacked in a cafeteria. Platters stored in the jukebox are known as "near on- line", and when a request is made to read a particular platter it is picked by a robotic arm, inserted into a drive, and made resident.

The control processor for a jukebox is typically a super- micro or mini-computer (e.g., a Sun workstation or a DEC VAX) with up to four jukeboxes under the control of a single control processor.

The retrieval times experienced for a "near on-line" pack is on the order of 10 seconds, while a resident pack may be accessed in about 200-500 MS. In addition to the robotic motion of 1-5 seconds, WORMs have spin-up, spin-down, and settling times of 4, 5, and 1.5 seconds respectively for twelve inch media, and 2, 3, and 5 seconds for 5-1/4" media. The steps required for configuration of jukeboxes will be familiar to capacity planners with mainframe experience. Essentially, we have an additional element in the memory hierarchy. After the number of platters has been determined, the ratio of resident drives to packs must be calculated based on access patterns. A typical strategy might be to configure a jukebox with four drives. Two might be made permanently resident, and used for storing new images, while two would be dedicated to servicing the near on-line packs in the jukebox. If a non- resident pack is requested (a 98% probability for randomly distributed inquiries based on 100 packs and two drives) then servicing the inquiry requires that one of the currently resident packs be removed and the new pack picked and made resident, a process which takes about 10-15 seconds. Ultimately, infrequently accessed packs will be stored totally off-line. Some implementations "copy over" a small number of still active files to an active disk before archiving the pack off-line, thus freeing up a juke box slot without a significant performance penalty.

PERFORMANCE IMPROVEMENT STRATEGIES

It is desirable to design a pack management strategy for a particular application which increases the probability of a request being serviceable from an already resident pack, since this pro-vides at least a twenty-to-one improvement in response time. In many applications, the probability of a retrieval request decreases with the age of the data. In one example for a claims processing system, it was determined that over 95% of the inquiries were for data under 90 days old. In such a scenario, it would be optimal to maintain sufficient resident capacity for data created in the last 90 days, even at the expense of not totally filling a jukebox to maximum physical capacity, thus ensuring that, in this case, 95% of the inquiries can be serviced with subsecond response time.

OPTICAL LOCAL AREA NETWORKS

We have noted before that even compressed images required 20-50 Kbytes each. Depending on the rate of image processing within the system, the capabilities of conventional local area networks (LANs) can be quickly outstripped. The best strategy is to draw the configuration and then determine the throughput rate (in images per second) which need to travel through the system. A network must then be constructed which can support this load.

Typically, the critical bottleneck is the jukebox control processor. Since each WORM drive can theoretically handle upwards of two images per second, and up to four drives can be installed in each jukebox, a four-jukebox system could be maximally processing 32 images per second, requiring a data rate of 1.6 megabytes per second.

A typical Ethernet, because of collisions and its small packet orientation (under 1,500 characters per packet), achieves a data rate of only 110,000 bytes per second in image-based applications, in our tests, even though it is nominally rated at 10 megabits/second. On the other hand, a token-ring, though nominally rated at only 6 megabits/second, can reportedly achieve a 60% utilization level (3.6 megabits/second) without excessive queueing delay. This rate is probably adequate for a single jukebox imple-mentation, but would present a bottleneck in our 4- box example.

Fortunately, fiber-optic local area networks have been introduced with nominal capacities of 80-110 megabits/second, which is more than adequate. Unfortunately, fiber optic LANs are still very expensive, priced at least five times the per-node cost of a typical conventional LAN ($1,000-$1,500 compared to $5,000 to $10,000). However, it is possible to intermix conventional and optical LAN technology in a network, using the broader band width only when required.

DISPLAY SCREENS

Designers of document management systems have benefitted from the desktop publishing revolution which has created tremendous demand for high-resolution, full-page display terminals. Display of an 8-1/2 by 11 inch document at 150 dpi resolution requires a display resolution of 1,275 dots per line and 1,650 lines, while 300 dpi would require 2,550 dots x 3,300 lines. At this writing, low cost (e.g., about $700) high-resolution displays are available at about 100 dpi, with 150-300 dpi displays expected to become available at moderate cost within the next twelve months. However, the memory required to support these high resolution displays will limit their use until operating systems become available which support the large memory spaces technically possible with the newer microprocessors (e.g., Intel 80386). In the meantime, 100 dpi is likely to remain the standard for page display. Optional software- or hardware-supported zooming features capable of showing a magnified portion of an image at a more readable 300 dpi are generally provided to overcome this limitation.

LASER PRINTERS

Typical laser printers can produce images at 300 dpi, although because of memory limitations some printers can image only half a page at maximum resolution. A point sometimes overlooked is that laser printers often communicate at 9.6 Kbs or 19.2 Kbs, which means that nearly ten seconds can be required just to transmit a full range image to the printer. However, special hardware boards are available which allow the laser printer to directly access the image in the microcomputer's memory, thus eliminating this bottleneck. Laser printers operating at speeds of up to 30 pages per minute are available, but the cost rises sharply above ten pages per minute. In addition, higher resolution (up to 600 dpi) printers are now becoming available. Duty cycles for laser printers must be carefully considered, since many are limited to 20,000 documents per month.

EXPERT SYSTEMS

Expert systems have many roles within document management systems. They are especially useful in the data validation phase. A general discussion of expert systems is, of course, beyond the scope of this paper. However, in sizing the work station, possible use of expert systems should be allowed for, suggesting the need for more powerful processors, additional memory, and a hard disk capability. It is also possible to maintain the expert system capabilities on a mainframe, but in many cases the required functions can be provided less expensively at the workstation.

SUMMARY

Optical technology, while posing new challenges for the capacity planner, builds on all of the accumulated experience gained in the last decade for mainframe capacity planning. As the volume of document management systems grows, we can expect a demand for more automated data gathering and modeling tools, but for now this is still a pencil and paper effort.

MAGNETIC CARTRIDGE TAPE CONFIGURATION MANAGEMENT

Anthony R. Guerrero
National Advanced Systems

ABSTRACT

If an organization decides to install a Magnetic Cartridge Tape Subsystem (MCTS), there are various aspects to consider in developing a good configuration and migration plan.

This paper explores the performance characteristics and capacity management needed to integrate this new subsystem into a computer complex. It also discusses tools and methods to assist installations in understanding their configuration needs and translating them into Magnetic Cartridge Tape units.

Other aspects are discussed, such as magazine loader use, productivity, operations, and Service Level management.

1.0. INTRODUCTION

This subsystem is the answer to the needs of today's Data Processing business. This new hardware architecture is a departure from the technology that uses 10.5 inch reels. It incorporates new features that have been needed to make magnetic tape a better price performer, but that were previously impossible to implement due to architectural restrictions. The architecture provides higher data transfer rates, datacompression/de- compression done through hardware, buffer/data management, and a much better media vehicle, chromium oxide tape, enclosed in a square cartridge.

Note: Refer to section 6.0 for a description of the hardware components.

Planning the installation of an MCTS requires a good understanding of the architectural implementation of the hardware functions and how they interface with other portions of the complex. In addition, the installation plan requires a working knowledge of the applications that will use the hardware. Completing the plan must include considerations in the use of Service Levels to better measure how well the equipment performs.

2.0 SUBSYSTEM OPERATION

There are some very important differences in the way in which the MCTS functions in comparison to the previous tape subsystem technology.

2.1 SUBSYSTEM FUNCTIONS

The control unit contains four megabytes of storage area which is shared by all drives. This area temporarily stores the read and write data as it goes to and from the processor to which the control unit is attached. The control unit architecture alleviates the effects of the tape mechanical motion by having the ability to disconnect from the channel while attempting to process a given request. In this manner, the MCTS is capable of reading or writing several blocks of data or servicing several requests, all in one tape motion cycle.

With such buffered storage, the control unit processes data in an overlapped fashion. That is, it is capable of transferring data from the buffer to the channel while it is transferring data from the buffer to a drive unit.

Tape buffering should not be confused with tape caching. The function of tape buffering, in the 10.5 inch reel technology, is to improve channel utilization by providing a means of balancing the faster channel speed with the slower tape storage device. On the other hand, in the cartridge tape technology, tape caching is the "intelligent" management of I/O operations and the transfer of data to and from the host to each individual magnetic cartridge tape. That "intelligence" also provides error recovery capabilities that are superior to the software initiated recoveries provided in the 10.5 inch reel technology.

The Subsystem functions in a Block Multiplexor Channel similar to the mode of operation for a DASD control unit. When the previous tape technology control units perform a function in behalf of a processor request, there is no disconnection until the tape drive sends a status on the completion of the request. This is the result of not having buffer and buffer management capabilities in the control unit.

With the introduction of buffer management, the control unit has the ability to disconnect from the channel immediately after the data has been transferred from the channel. And, by improving the control unit and tape drive speed for data transfer, the data exchange is no longer bound by the limitations of the control unit or tape drive, but by the speed of the channel.

2.2. SYSTEM ALLOCATION

The Subsystem can be attached to more than one Central Processing Unit. It may be "configured" through the VARY command, which assigns and unassigns the number of drives in the configuration for each processor.

While earlier tape subsystems obtained drive assignment through operator intervention (a mixture of software commands and manually thrown switches), manual switches are not used in this hardware. This is because of the large number of switches that would be required to accommodate all the possible combinations from eight tape units, one control unit and four channels, to the possibility of sixteen tape units, two control units, and eight channels (Dual Controller Coupler option).

2.3. SIMULTANEOUS DRIVE OPERATIONS

The architecture of the subsystem permits the overlap of operations. In other words, the control unit can transfer data from one drive to the processor while it sends data to another drive from the processor.

2.4. DRIVE MAINTENANCE

The required drive cleaning has been drastically reduced due to the enclosed nature of the hardware. The recommended drive cleaning schedule for an environment with low tape utilization is one time per week. This will help productivity when it is done on "scheduled time" rather than when it is actually needed.

When tape drive activity is heavier than anticipated, and cleaning is required before scheduled time, the hardware will display a "CLEAN" message. This signifies that the cleaning cartridge must be inserted in the drive to remove the

accumulations from the Read/Write assembly, and will be displayed after a drive has processed at least 50 cartridges. The tape drive cleaning cartridge contains non-abrasive material which removes the residue deposited by cartridge activity.

3.0. CONFIGURING THE SUBSYSTEM

Data centers have often planned and installed excess configuration when using previous technologies. This is due partly to the difficulties related to architecture, reliability, performance and capacity limitations that the hardware had to offer. In addition, the priority focus on growth management has always been on DASD. Magnetic tape cartridge technology has increased the need to develop a consistent methodology for configuration and growth management regarding tape drive units.

To properly configure each subsystem string with an adequate number of tape drive units, the number of concurrently active drives needs to be established. It is possible that with multiple active drives the effective data rate can decrease. This situation is created by sporadic interruption in drive data streaming, buffer management overhead, or as you add drives, an increase in buffer management and data transfer activity. It is also important to remember that when a drive is not transferring data, but still active (rewind, forwarding, etc.), the control unit cannot transfer any data until that particular activity finishes.

The maximum string configuration for concurrent drive operations is 4 drives units, or up to eight drives per subsystem. But, a general guideline cannot represent sound business, unless it bridges the need to maintain an acceptable high level of service. For a dual coupler MCTS, the above still holds true except that the dual coupler will be capable of doing buffer and I/O path load balancing. This is especially important when configurations are not properly balanced in terms of the number of drive units attached to each control unit (i.e. four drive units in one string and two drive units in the other).

3.1. THE CONFIGURATION TOOL

The first step in determining what the configuration for the new subsystem should be is to analyze the SMF data for the system to which the tape subsystem will be attached. This analysis can be done by time periods, and at least, for each hour within a shift. This will give a better understanding of the real utilization of a present tape drive configuration.

Furthermore, the installation needs to identify the maximum number of concurrent drives required at a given time. This can also be extracted from SMF and confirmed through system log by the messages issued by the operating system whenever a mount cannot be satisfied due to unavailable drives .

The analysis has to represent the view of the installation regarding Service Levels policies. The data must represent the number of tape mounts to be satisfied (per hour, per shift, etc) in conjunction with the tape usage and identified in hours (or percentage) for the same period.

The data relates to:

- Single volume tape datasets
- Number of records per dataset
- Format, record and block sizes
- Number of drives allocated
- control unit busy
- Tape drive unit busy
- Number of scratch tapes used

The data is found in SMF records type four, five, fourteen, fifteen, thirty and seventy-four. Algorithms are provided to determine the recommended number of drives, cartridges and magazine loaders for a specific environment.

The tool is composed of two phases. The first phase extracts the necessary data to be sent to a personal computer from the SMF records. In this phase, there is also a description of the present physical configuration of tape subsystem(s). It includes pathing data for each tape drive and control unit that are part of a subsystem. This phase executes as a batch job in a host CPU. Its data reporting/extracting capabilities are for time periods or specific jobs. It is also capable of extracting data by shift or work day.

The second phase, which runs in an IBM-compatible personal computer, produces the subsystem, cartridges, and magazine requirements for a particular installation. The presentation function produces reports and graphic output. There are also performance and modeling capabilities to permit further analysis of the proposed configuration.

The entire process should take no more than two hours of a person's time.

4.0. SUBSYSTEM PERFORMANCE

It is very important to have a plan in place to monitor the performance of the subsystem. One of the key areas to monitor is utilization.

4.1. SUBSYSTEM UTILIZATION

An important item to gauge is the ratio of logical control unit disconnect to connect time. To look at percent of channel busy could be misleading, since most of the subsystem disconnect time is spent in the channel, waiting for data transfer to complete from another drive unit. Consequently, when the time spent in transferring data (connect time) is exceeded by the time spent waiting for the control unit to become free, the need to add new data paths becomes important. Situations like buffer-full conditions would reflect here as part of the disconnected time of an I/O operation. However, since the subsystem will be used by batch jobs in the majority of the time, this situation needs to be evaluated in the light of how important it is that the work be completed within a given time window.

5.0. SUBSYSTEM SERVICE LEVELS

Service Level management is the methodology that provides a vechicle for users to communicate their needs to the Management Information Systems (MIS) organization. Also, it allows MIS to respond to those needs by setting objectives in various areas, such as, performance, availability, capacity, reliability, security, and accessibility.

The MTCS poses a new requirement within the Service Levels methodologies. With the advent of Systems Managed Storage to Data Processing, the approach of Logical Service Level management requires a unique view of every cartridge volume and its data content.

5.1. LOGICAL SERVICE LEVELS

Logical service levels define the performance, and availability objectives for a given application.

Those tape cartridges which contain the files for an application must be part of a Logical Service Level agreement. There can be one agreement per application which may contain multiple volumes.

Since a volume can be used by multiple applications, there can be multiple logical service level agreements for a given volume.

Most applications require that their data be readily available and accessible when needed--thus, the basis of a Logical Service agreement per application. Each agreement must contain one or all of the three elements that cartridge tapes should "live" by: availability, accessibility, and reliability

5.1.1. **Availability and Accesibility.** Although there can be many ways to implement this measurement, the most important choices are based on: a) the amount of time it takes to process a mount request. (This would include the amount of available drives at a given scheduled time.); and, b) how long it would take to get a new copy if a dataset or volume had to be recovered.

5.1.2. **Reliability.** In this category, it is important to understand that the user is dependent on the technical input from Data Processing. However, this is a management gauge to determine how well the maintenance procedures and media quality control methods are working. Some of the items in this category are: a) errors per processing run b) permanent errors per volume per run c) number of abends and reruns caused by I/O errors atributable to tape maintenance or media quality

6.0. SUBSYSTEM DESCRIPTION

This subsystem is the product of the technological advances made in the last twenty years. Its design incorporates the needs of the DP community and the knowledge gained from Large Systems in the last ten years.

6.1. THE CONTROL UNIT

There is one control unit for the Magnetic Tape Cartridge Subsystem. Its standard features are one channel interface and a data rate of three megabytes. Options include six megabytes data transfer rate, data compression, and up to three additional channel switches. Additionally, another control unit with eight tape drives can be connected to the primary control unit.

Each subsystem can have up to eight drives (four drive units) attached to each control unit. When paired with another control unit, this number increases to sixteen drives (eight drive units).

6.2. THE DRIVE UNIT

The drive unit has two drive transports as a standard feature. Optionally, it can have an automatic cartridge magazine loader and an eight-cartridge magazine. The magazine is a productivity enhancement. Its rack mount design permits up to eight cartridges to be automatically fed into a tape drive.

A tape drive can rewind a cartridge tape in approximately forty-eight seconds. Its load/unload time is approximately seven seconds, plus an additional 1.5 seconds for insertion or extraction from the unit. It moves the tape at 78.7 inches/second (two meters/second).

Each tape unit contains a microprocessor which makes new functions available. While these functions are being performed, the drive is disconnected from the control unit, thus enabling its control unit to perform other functions in overlap.

A display panel on top of each tape drive is used by the hardware and the software for messages regarding volume activity, errors and drive maintenance.

6.3. THE TAPE CARTRIDGE

The tape cartridge can contain approximately 200 megabytes of data when written in 24k blocks. This data is stored in 18 tracks at a density of 38000 bytes/inches. The cartridge dimensions are approximately 4" x 5" x 1". Its size is approximately 75% smaller than a 2400" tape reel.

The cartridge is very simple to use. To load it, all the operator needs to do is insert it in the drive, and the tape will be threaded automatically.

It is not necessary to place reflective markers at the beginning or the end of the tape. This need is eliminated because the drive, in itself, determines the beginning or the end of the tape. Since the tape is enclosed within the cartridge, it provides increased protection from physical damage and other environmental factors. The mechanism for file protection is a "thumb wheel" selector which is integrated in the cartridge.

7.0. SUMMARY

This subsystem presents new technological features that will ease tape management requirements and maintenance. At the same time, it presents some options that will increase people productivity at the operational level. The real challenges still reside in the capacity planning and performance management, where there is some work to be done.

8.0 REFERENCES

1. NAS Publication FE-97TD400, The 7480 Cartridge Tape Subsystem Physical Planning Guide.

2. NAS Publication FE-97TD403, The 7480 Cartridge System User's Guide.

3. IBM Publication GA32-0041, IBM 3480 Magnetic Tape Subsystem.

END-USER PERFORMANCE MEASUREMENT: AN AUTOMATED PC APPROACH

Gerri K. Martin Flickinger
Chevron Corporation

ABSTRACT

Measuring true end-user response time in complex networks has been difficult in the past. One possible solution uses a PC to emulate user activities to gather response and availability measurements. The merits of this measurement method and how it compares to other network performance measurement tools, will be discussed. Chevron's experience with a PC-based system (benefits, installation, automation) at many sites within a large IBM SNA network will be shared.

1. BACKGROUND: END-USER MEASUREMENT

Past attempts to measure end-user response time and availability have proven difficult. True end-user measurements are extremely useful for service level agreements and performance analysis, however, such measurements can be expensive and technically complex. Following is a discussion of the motivation for end-user measurements and measurement techniques currently available.

Motivation for end-user measurements may come from several sources. There are user concerns that availability and response experienced at their sites' differ from that reported at the central data center. In addition, there are concerns that benchmarks run in the host environment are not comparable to end-to-end service levels. This leads to the concern that service level agreements do not sufficiently account for variation in user location. Finally, Performance Analysts often find they lack accurate information on the impact of hardware and software changes to the end-user.

There are several methods of capturing end-user response information. Some methods use VTAM definite response produced for some applications while others monitor traffic across a certain network component.

If a measurement is taken at the host, a message can be time stamped when it enters the system, time stamped when it is outbound after processing and time stamped when the definite response returns from the control unit as requested by VTAM. This method provides both host and network response time information. Many applications, however, do not use definite response so this measurement technique is not available for these. Examples of products that perform this type of measurement are NPM [1] and MAZDAMON [2]. (See Figure 1.)

If a measurement is recorded at the control unit level, a time stamp is taken when the message is inbound (to the host) and when the response is returned to the control unit from the host. This method only gathers end-to-end response time as far as the control unit. This method is not transaction-oriented, therefore the host response may not be at the logical completion of the applications transmission. Furthermore, there is significant host overhead involved in collecting all the response information from the control units. An example of this type of product is RTM [3]. (See Figure 1.)

Another method is to use a hardware device directly attached to a user terminal that records all terminal traffic. One problem with this method is that the user must make an effort to upload or send the data to a central site for processing. Furthermore, if the user is not busy, the device will not be collecting data even if other users on the control unit are working. An example of this type of device is a TEMPO monitor [4]. (See Figure 1.)

Other methods include hardware monitors on various network components (e.g. RTA [5]) or user questionnaires. Many of these methods are difficult to maintain, only capture a portion of the end-user response measurement or are subjective in nature.

Most of above methods have a major shortcoming; the measurement of availability. Measurements based on definite response or terminal monitors require that user work is being done. Not only does this require that a user is active, it also requires that all network components (e.g. control unit, NCPs, etc.) are available. Since availability is a critical component of end-user performance, these methods do not satisfy the entire requirement.

Another concern with these methods is their level of complexity. Generally a Senior Analyst is required to install and maintain such products. There are dependencies upon host and network internals. Since the data is not transaction oriented, its interpretation is frequently complex. Furthermore, these methods can be very CPU intensive and expensive to purchase and maintain.

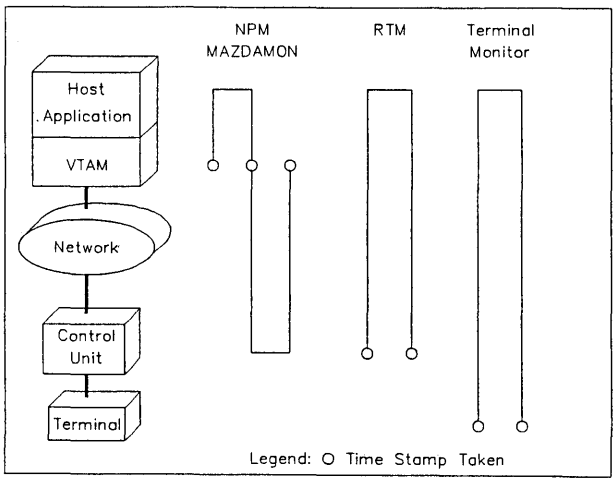


Figure 1. Measuring End-User Performance

2. PC-BASED USER EMULATION APPROACH

An alternate approach is to use a PC that emulates a user. The PC logs on to user applications, performs user-type functions and measures response and availability. Since the PC is independent of the host, network and users, it can reliably record outages as well as response times.

One major advantage of this solution is the increased degree of flexibility. A single PC can test access to several different hosts and applications without regard to the internals of the systems. This is extremely important when determining the delay caused by different components of the network. For example, placing several PC monitors in carefully chosen end user locations, and testing access to a single host, will measure response time on several unique paths to the host. Extremely complex networks can be monitored efficiently because of the flexibility of access.

The PC solution is far simpler since it deals with an end-user viewpoint and does not require special software, hardware or specially coded applications in the host or network. Implementation of such a monitor and interpretation of the data collected does not require an in depth knowledge of host, network and application internals.

Since the PC is performing user-type work from a user perspective, it is transaction oriented. Since the transactions are chosen in advance, consistent benchmarks are available for site comparisons. By placing several PC monitors at key locations in the network, routing and cross-system delays can be approximated.

The PC can retain the data for a period of time and then upload to a host for analysis, thus running unattended and without human intervention. The host overhead associated with each PC is comparable to an additional user on the network. Because of the flexibility in choosing the specific transactions to be measured, the storage requirements and reporting effort needed for this method is far less than for other end-user monitoring methods.

There are several products available that provide a basis for PC user emulation monitors. These products generally provide a script language for specifying keystrokes and recognizing host responses. An example is PECOMON [6] which has a script language that consists of several look-up tables. Another product is MORTII (Monitor of Online Response Time) [7]. MORTII (later referred to as MORT) provides a high-level script language as well as additional features such as global recognition (e.g. if you EVER see a lightning bolt...) and flexible recording capabilities. MORT uses an IRMA 3278/79 emulation card [8] to communicate with the SNA network.

3. CASE STUDY

In response to Chevron's need for end-user measurements, and in light of the shortcomings of previously used methods, a test of MORT in a production environment was planned.

3.1 CHEVRON'S ENVIRONMENT

Chevron's environment under study consists of two major data centers located in the San Francisco area. These data centers support all of the corporate and marketing data processing for Chevron. There are 10 CPUs running 14 system images (MVS and VM) between the two centers. The CPUs consist of 2-IBM 3081s, 7-IBM 3084s and 1-IBM 3090-200. Chevron's SNA network includes approximately 20,000 terminals, and 30 IBM 3725 communications processors, and will soon include about 30 IBM 3708s.

Since numerous hardware and software components may be involved in a single transmission from a terminal to a host application (e.g. possibly several NCPs, CPUs, telephone lines, cross-domain MSNF, etc.), it is understandable that end-to-end response time and availability at a remote location may be quite different than at the central site. Furthermore, many locations perform different computational functions on different CPU complexes, making the establishment of a single benchmark difficult or impossible.

3.2 MORT INSTALLATION

In February of 1986, Chevron began a five month Pilot Project. During this time, MORT was installed at three sites in the San Francisco area and Chevron was able to:

- operate three sites 24 hours/7 days week without human intervention
- validate MORT data
- centralize data reporting
- automate update capabilities
- develop methods for communicating with MORTs directly from the central site
- write 20 different scripts tailored for various user functions and performance measurement requirements
- automate portions of our reporting mechanisms
- provide near real-time analysis facilities
- provide a way to receive "network time" (including VTAM) in addition to end-to-end time.
- develop a plan for PC management
- respond to security concerns

At the end of the Pilot Project, Chevron found that MORT could provide a basis for a powerful and reliable monitor system. Additional sites were recommended that would incorporate most of the remote NCP locations.

Today Chevron has 18 MORTs installed. Sixteen of the sites are considered "permanent". Two of the sites are "portable" and are moved to trouble spots in the network to help diagnose or verify problems.

The following section discusses several key components of the installation and design decisions explored.

3.3 RUNTIME ENVIRONMENT

MORT, PECOMON or any other script language simply provide a means for writing user scripts and not necessarily a complete response monitoring environment. The final monitoring environment at Chevron contained several software components in addition to MORT scripts. Each component is discussed below.

Many scripts were written to address various user applications and host benchmarks. For the VM environment, scripts were developed for PROFS, PL/1 compiles, NOMAD applications and trivial transactions. These scripts are representative of end-user work, such as the PROFS script which logs onto PROFS, writes a note, sends it, opens the mail, deletes notes and logs off. Scripts have also been developed for the MVS environment, specifically for IMS and CICS applications.

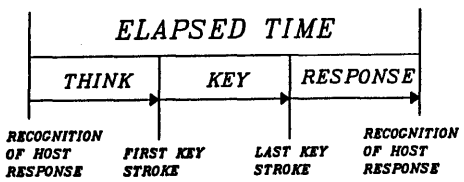
All scripts require coding of keystrokes for normal as well as abnormal sessions. The majority of programming time is spent in coding error recognition and recovery rather than normal session keystrokes. An average script is eight to ten pages, 90% of which is abnormal session recognition.

In the case of the MORT script language, the decision of what portion of the session to gather response information on is left up to the script programmer. At any point, timers may be turned on or off, and multiple (overlapping) timings are possible. An example of this is shown below in a section of a MORT PROFS script

```
on <R; T>,*,1   type <PROFS[ENTR>
                begin transaction <PROFS-BENCH-TOTAL>
                begin transaction <PROFS-BENCH-MENU>

** when we see main menu stop the MENU timer

on <A00>,1,78   clear transaction <PROFS-BENCH-MENU>
                type <[PF4>
                      :
                      :
** perform all PROFs work indicated and then exit
** PROFS and stop TOTAL timer when we are in CMS

on <A00>,1,78   type <[PF12>
on <R; T>1,78   clear transaction <PROFS-BENCH-TOTAL>
```

There are two timers in the above example: PROFS-BENCH-TOTAL and PROFS-BENCH-MENU. When the data file from this script is written out, there will be 1 record for each timer and each record will include an end-to-end response time for the portion of the session timed.

One comment should be made about end-to-end times gathered by MORT. MORT is very careful not to include his own time to type keys or execute script statements into the response time. MORT timings are subdivided as follows:



For the purposes of end-user response measurements, we are only concerned with response time as shown in the diagram above.

A fundamental requirement for a 24 hour, stand alone monitor, is the ability to upload data to a central site. MORT provides an upload facility, although it is extremely slow. A different approach is to use the standard SEND function provided with the IRMA emulator.

Another requirement is to be able to run different scripts at different times of the day and different days of the week.

A reasonable solution that addresses both of the above requirements is to code an EXEC on the MORT PC that is responsible for executing the various MORT scripts at the appropriate times and issue the proper send command at certain intervals to perform uploads. The EXEC (later referred to as the shell) also provides a nice framework for invoking different scripts from different sites rather than writing one large script, with multiple system tests, for each site. Specifically, the shell invokes individual scripts (for example a PROFS script for a specific VM host, an IMS script to test a specific application,etc.) and each script is completely autonomous. Using the shell for overall control insures that the runtime environment is consistent for all sites even though completely different response measurements (scripts) may be taken. From the time the MORT PC is IPLed, the REXX shell is in constant control.

The PC EXEC language chosen for this implementation was Personal REXX [9] because of its power and similarity with REXX on VM.

In addition to end-to-end time, a method of measuring network and host time was desired. Although a PC-based monitor is not designed to gather these figures because of its simple-minded approach and physical location, there are ways to gather these values. One way to perform this for a VM transaction is to invoke the desired transaction through an EXEC that takes time stamps on entry and exit. For example:

```
/* Invoke a VM benchmark, for example       */
/* enter and exit XEDIT.                    */
/* Time this benchmark and print the result.*/

T=TIME('R')
PUSH QUIT
'XEDIT GARBAGE DATA (NOPROF'
SAY 'RESULT: 'TIME('E')
```

(Note that the choice of REXX over EXEC2 in the VM environment provides additional precision.) If the value from this EXEC is retrieved by MORT and recorded with the benchmark end-to-end timing, we can subtract the two values and receive an approximation of network time (including VTAM overhead).

Security was also a concern for each MORT PC and all the associated host IDs. Since the PCs contain scripts with confidential information (IDs, passwords, and procedures), each MORT PC is secured in a lock box that contains the system unit as well as keyboard thus preventing unauthorized access to the information. Unauthorized access of the IDs can be monitored by a MORT script itself. For example, when a script fails a logon because the id is already

active on another node (LU), the script retrieves the LU name from the screen and records it in the data file. A listing of all security breeches is produced daily as part of standard reporting.

A final, yet crucial, concern with any monitoring system is the cost. This includes the one time cost for equipment and software (see Appendix I) as well as runtime costs, disk space and manpower. Considerable cost analysis was performed before placing MORT in production. Following are some considerations:

Cycle Time: how often does each script need to run? The Chevron installation determined every 10 or 30 minutes, per script, per site, based on application.

Time of Day: what time of day (and day of week) do we need to test? This is based on scheduled application hours at Chevron.

Disk Space: what timings do we need? Remember each timing cuts a record. The Chevron installation found that individual transaction timings are generally not necessary, but rather timings for a set of transactions that make up a meaningful work unit (e.g. logon, order entry).

Manpower: how much manpower are you willing to spend? For each MORT site and each script there is manpower to monitor the MORT, diagnose problems, run reports and analyze data. The manpower required for the initial installation and ongoing support is critical to providing the most useful and effective MORT data. Because of this, sincere management support is crucial for such a project.

The PCs used for the MORT sites are IBM PCs with 640K of memory. Hard disks were added to all PCs to allow file space for multiple scripts (approximately 19K per script), the REXX shell (12K), additional software such as the MORT interpreter, IRMA Terminal Emulator, SEND, Privacy Plus and Personal REXX (approximately 470K) and data (approximately 13K per hour). There was no need to run the MORT monitors on any larger or more expensive machine. PC displays, although not necessary for the operation of MORT or any other software used, are included with the PCs for the analysis or trouble shooting of MORTs.

Although the Chevron Installation used VM IDs for data collection and reporting, the MVS environment would also provide the same flexibility.

3.4 MAINTENANCE AND SUPPORT

Immediately it was realized that with 20 unique scripts and 18 sites, (over 150 scripts total) maintenance needed to be automated. The solution was to write a series of programs that would modify scripts and generate new ones automatically. These programs were written' in Personal REXX and depended upon macros written for KEDIT [10] (the PC equivalent of XEDIT). These execs reside on a PC/AT located at the central site where MORT maintenance is performed. The entire set of routines is approximately 35K.

The first step to automating maintenance was to develop script templates for each application. These templates do not include system, id or password information. After these are in place, a program can be run that leads the user through script setup. The user specifies the desired MORT number (a unique site identifier) and desired system. A new script is generated using the template, the user site information and id lookup table. Following is one' of the menus generated by the setup routines.

```
---------- M O R T   S E T - U P ---------------
------- (property of Chevron Corp.) ------------
--------------------------------------------------
------------- MORT number: 01 -------------------
----- Directory for new files: C:\MORT\M01 -----
--------- Set-up type : new site ---------------

      The following are valid scripts:

VM       1.........PROFS benchmark
         2.........CASE benchmark
         3.........SCORE benchmark
         4.........VM short benchmark

CICS     5.........Credit Card benchmark

IMS      6.........COS Well Inquiry
         7.........CPS Bulk Order
         8.........CPS Marketing
         9.........CHEM Order Entry
        10.........CHEM Additive Order
        11.........CHEM Accounts Receivable

UPLOAD  12.........LOGON to upload account
        13.........LOGOFF to upload account

         0.........EXIT (no script set up)
```

After generating a script, the problem of how to get it to the remote site was raised. The first solution was to write the script to a floppy, mail it, wait for the site to receive it and wait for someone to load it onto the PC. A faster method is to send the update through a dial-up connection or the existing network to the remote PC. Both of these options were explored.

Dial-up access was attempted using Carbon Copy [11], a PC-based asynchronous communications package that allows one PC to take control of a remote PC to send files, etc. Unfortunately, Carbon Copy and Personal REXX both utilize the same interrupt vectors so they cannot run on a PC simultaneously. An additional concern with any dial-up access method is security. Furthermore, a modem is one more piece of equipment to purchase, maintain and troubleshoot.

The second alternative, sending files through the already existing network was developed. The new script is generated on a PC at the central site. It is uploaded to a VM host and sent to the upload (data accumulation) id of the destination MORT. When the MORT PC logs on to upload response data, an EXEC on the host receives files from the reader onto the host id. The shell on the MORT PC receives the updated script and compiles it. If a new shell is sent, the current shell receives it and loads the new shell into memory destroying itself. Any file can be sent using this method. The entire process is automatic from the time the upload is requested on the central site PC. (via a menu option)

An additional safeguard of encrypting all scripts is in place. Encryption is done at the

central site PC after the script is generated, and de-encryption is performed by the MORT PC at the remote site. Again, the entire process is automatic, and no human intervention is required. Privacy Plus [12], a PC-based encryption scheme, is used for this process.

3.5 DATA REPORTING

Data reporting on end-user response data as described above is primarily a post-analysis process. Based on cost analysis and the impact on the network of data transmission, MORT data is uploaded every 1/2 hour. This provides an opportunity for data analysis within a 1/2 hour of data gathering.

An EXEC runs on the VM host every 1/2 hour that checks data from all MORT sites. It scans the data for outages, high response times, abnormal situations or missing data. The EXEC produces a log summarizing these events and also sends messages to appropriate Performance Analysts for investigation of notable conditions (including maintaining MORTs operation). This method provides near real-time analysis (1/2 hour old data). Following are some examples of the type of warnings generated and sent to appropriate analysts. (See Example 2.)

On a daily basis, all MORT data is read into SAS datasets for consolidation and reporting. Daily online reports are available to all PROFS users an example of which is shown in Figure 2. In addition to the daily online reports, monthly reports for management are produced. These reports include trend analysis and discuss any unusual or interesting response characteristics of our network, hosts or applications.

A variety of data can be extracted from the MORT data gathered. Figures 3 - 6 contain examples of graphs and plots that illustrate the diversity of some of the data.

```
16:00 MORT 05  HIGH  Host Response, COVMA : 0.163017 at 15:32:06
16:00 MORT 07  HIGH  Host Response, COVMA : 0.153470 at 15:30:45
16:00 MORT 08  8 occurrences of No VTAM Banner since 15:30
16:00 MORT 11  HIGH End-to-End      CPVMB : 10.055 at 15:44:01
16:00 MORT 15 12 OUTAGE records, Chem/SFIMSP since 15:30
16:00 MORT 15  8 OUTAGE records, COS/SFIMSP since 15:30
16:00 MORT 16  NO DATA SINCE 9/3/87  14:34:29
```

Example 2. Near Real-Time Alerts

```
                    Performance Management Online Reports
                          MORT VM Availability
                            09/03/87  (prime time)
                    Number of attempts  :  % successful
SITE       CPVMA      CPVMB      CPVMC      COVMA      COVMB
ChvPark   46 100%    46 100%    46 100%    46 100%    46 100%
BR6       51 100%    52 100%    51  98%    49  98%    50 100%
Concord   36 100%    36 100%    35 100%    35 100%    35 100%
Hilltop   36 100%    36 100%    36 100%    36 100%    36 100%
CRC                  53 100%               53 100%    53 100%
GulfTwr              52 100%                          52 100%
Algiers              38 100%                          37 100%
Tampa                45 100%                          45 100%
SanFran              49 100%    49  98%               49 100%
Towson               50 100%                          50  98%
Fairfax                                               44 100%
Oklahom                         40  97%    38 100%    39 100%
Hobbs                43 100%    42  98%    43 100%    43 100%
Bakersf                         38  97%    37 100%    37 100%
Houston
Tulsa                54 100%                          53  98%
 PF3  End    PF7  Next Day    PF8  Prev Day    PF9  Help
====>
```

Figure 2. Daily Online MORT Reports

4. BENEFITS REALIZED

End-User measurement has some obvious benefits in isolating components that cause outages or degrade response. End-user emulation has provided a unique perspective into the performance of Chevron's host, network and applications. Following is a discussion of some benefits of end-user measurements that have been experienced at the Chevron Installation.

4.1 Technical Performance Analysis

MORT is surprisingly useful in testing new hardware (e.g. IBM 3090, 3174, channel extenders, LANs, etc.). As an example, during a period when channel extenders and 56KB line performance was being compared, MORT showed some very convincing response time figures in favor of the channel extenders.

A related use for MORT is in analyzing the impact of new software, on the end-user. Recently Chevron installed VM/VTAM on all their VM hosts (replacing VS1) and MORT data was used to verify that there was little user impact caused by the change in network software. Of course there are other sources for some of these measurements such as the use of TPNS [13], although such methods only simulate an end-user environment.

Another area MORT data has proven useful is network configuration. MORT data helped alert and isolate several performance problems in our network related to network configuration. One such occurrence was caused by unequal loads on multiple gateway NCPs, causing certain users (and MORTs) to experience very degraded response, while providing other users (and MORTs) excellent response. Once the loads were balanced, the response time for all users across the gateway stabilized an uniform level.

A similar problem occurred with NCP virtual route definitions that were set too low. Certain locations experienced long periods of unavailability when trying to cross this particular gateway. After the limit was modified, access across the gateway was no longer inhibited from any site. The data that MORT gathered, from its various network perspectives, has helped alert, verify and isolate this type of problem.

The routine tuning changes that are frequently made to improve system performance, can now be viewed from the end-user perspective. Performance Analysts, after making a system parameter change, often review MORT data to ensure that there has not been an unexpected impact on end-user performance.

Frequently, when responding to serious user complaints about response time or availability, MORT data is used to support the claim and help isolate the problem to a data center, host, application or network component.

Surprisingly, using a PC as a "perfect user" (i.e. the PC does not hit the wrong key) we have found some deficiencies in some of our applications. We have isolated several problems that were attributed to "user-key" error in the past. When MORT nodes were

traced, we were able to verify (from a host perspective) that the PC was sending the correct characters, and the problem was within the application.

Besides these technical and support uses for MORT data, we find that ordinary outages can be verified. Occasionally, the duration of an outage at the data center may be considerably different from that felt at remote sites. MORT data alerts us to procedural problems in recovering from outages at the central site; sometimes we inhibit remote users for longer than is necessary.

Utilizing the near real-time alert method described in Section 3.5, Performance or Operations Analysts can be notified of interesting or unusual data so that corrective action, if necessary, can be taken as quickly as possible.

4.2 Service Level Agreements / Capacity Planning

In our design of MORT scripts, we attempted to provide two fundamental classes of data:

BASELINE BENCHMARKS

- Benchmarks useful to Performance Analysts for network-wide evaluation.
- Benchmarks that are identical for all sites.
- Provide end-to-end, network and host time.

APPLICATION / USER-GROUP BENCHMARKS

- Benchmarks "meaningful" to users (i.e. user-type work, CICS and IMS applications, PROFS, etc.).
- Benchmarks that vary from site-to-site based on business function.
- Generally provide only end-to-end time.

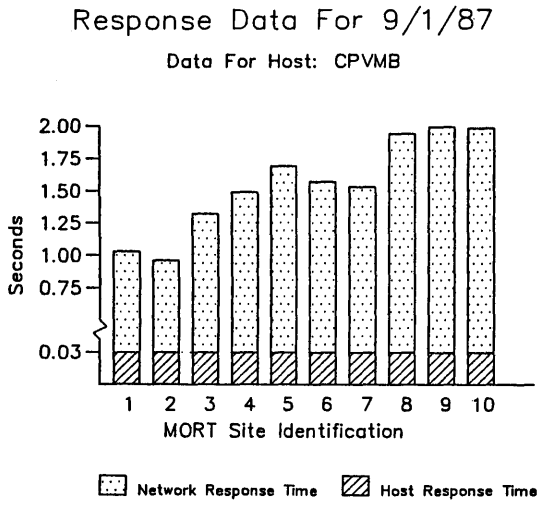


Figure 3. Host and Network Response Time

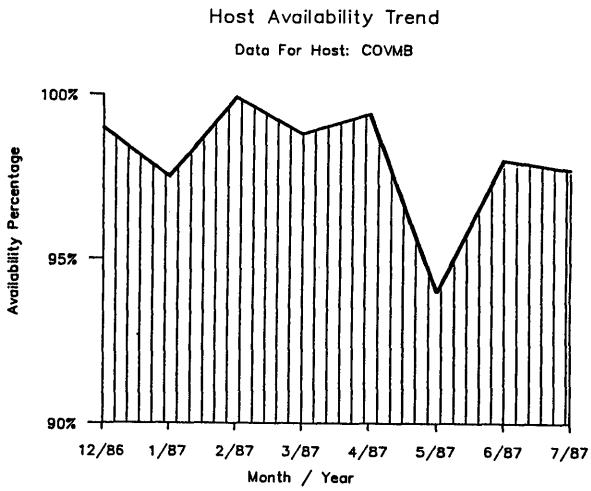


Figure 4. Availability Trends

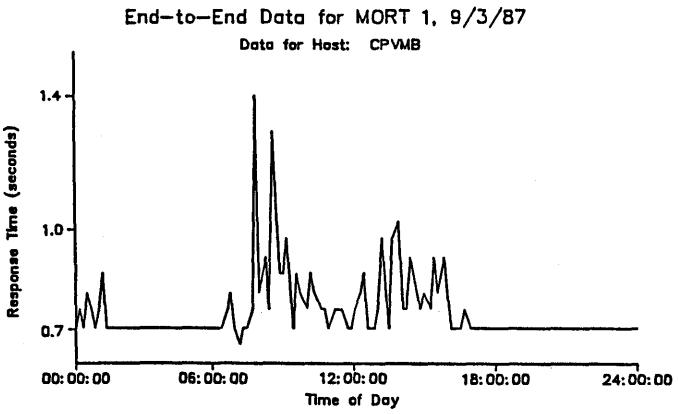


Figure 5. Daily End-to-End Response Curve

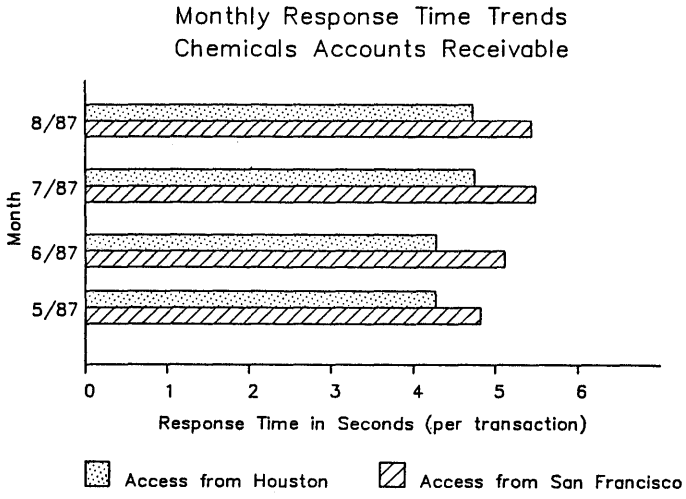


Figure 6. Application Response Time Trends

MORT is a unique tool that can help determine the current levels of service we are providing based on location and specific application as well as generalized service levels computed from baseline benchmarks (e.g. what is the impact on response time of 1 NCP? 2? etc.). Both of these measurements seem valuable for Service Level Agreements. It is important to have benchmark figures that management and users can equate to "real work" at their sites and it is also important to have benchmarks that provide comparisons between all sites. Both of these needs can be fulfilled with the type of scripts we have designed and implemented.

As the load on the network and hosts change, we can also view the impact on end-user performance. This can be very useful for capacity planning. Although products such as IBM's TPNS provide a host-based stress test for capacity planning, MORT provides a measurement of true end-user response time. The analysis of MORT data can be valuable in determining at what point increased capacity is needed based on end-user response and availability requirements.

5. CONCLUSION

MORT provides advantages that some other end-user measurement methods lack. Availability at the end-user location can be measured. MORT does not require a Senior Analyst with extensive knowledge of host, network or application internals, to be effective. MORT has very flexible measurement methods and a high-level script language. Overall, MORT provides a very straight forward solution to a historically difficult problem.

The Chevron installation provides a complete end-user response and availability monitoring environment. Concerns of script generation, data gathering, site maintenance, automated reporting, and trend analysis have been addressed. The entire system relies upon several software components including REXX, MORT, Privacy Plus, KEDIT, and REXX on VM. Several aspects of the maintenance and runtime environment have been automated including script generation, update distribution and near real-time alerts.

Placing PC monitors at key user locations can provide end-to-end response and availability data vital for technical performance analysis, service level agreements and capacity planning.

Overall, MORT has been very useful in analyzing the data processing services we are providing the end-user. No other monitor tool has been able to provide the same unique perspective into our network, host and application environment.

6. REFERENCES

[1] NPM (Network Performance Monitor)
IBM
Contact you local IBM Representative.

[2] MAZDAMON
Computer Associates
2195 Fortune Drive
San Jose, CA 95131

[3] RTM (Response Time Monitor)
IBM
Contact you local IBM Representative.

[4] TEMPO Monitor
DTSS Inc.
10 Alan Street
Hanover, NH 03755

[5] RTA (Response Time Analyzer)
Datacom Management Systems

[6] PECOMON
IBM
Contact you local IBM Representative.

[7] MORTII (Monitor of Online Response Time)
Automation Design Inc.
350 N. Clark, Suite 650
Chicago, IL 60610
(312) 670-2660

[8] IRMA 3278/79 Emulator
Digital Communications Associates (DCA)
1000 Alderman Drive
Alpharetta, GA 30201

[9] Personal REXX
Mansfield Software
P.O. Box 532
Storrs, CT 06268
(203) 429-8402

[10] KEDIT
Mansfield Software
P.O. Box 532
Storrs, CT 06268
(203) 429-8402

[11] Carbon Copy
Meridian Technology
1101 Dove Street
Newport Beach, CA
(714) 476-2224

[12] Privacy Plus
United Software Security Inc.
McLean, Virginia 22101

[13] TPNS
IBM
Contact you local IBM Representative.

APPENDIX I

Approximate hardware and software costs for a single MORT site at the Chevron installation.

IBM PC or XT with Hard Drive	$ 3000
Monochrome Display / Adapter	300
Memory Expansion to achieve 640K	250
MORT	1090
Security Box	170
Privacy Plus	180
Personal REXX	50
	$ 5040

INCORPORATING DATA NETWORKS INTO A PERFORMANCE AND CAPACITY PLANNING STRUCTURE — A CASE STUDY

Hugh Smith and Dyna Wilson
CoreStates Financial Corporation

ABSTRACT

Given the growth in the number and importance of on-line applications, one can no longer afford to ignore the impact of data network performance and availability on end user service. Internal response times and host/application availabilities alone are not sufficient to identify or to predict performance problems in an on-line transaction environment. It is imperative that end user service levels include the impact of data network performance and availability.

A case study is presented in which data network issues are incorporated into an existing Performance and Capacity Planning structure. The environment consists of an on-line transaction volume of 1,350,000 transactions daily originating from more than 8000 terminals, ATM's and point-of-sale devices.

Central to the structuring of network issues is the recognition that two distinct areas must be addressed: Availability and Performance. Although each of these areas requires monitoring, reporting and planning functions, the responsibility for these functions may be best suited for different organizational elements within an institution. A methodology for the structuring of network performance and capacity planning is evolved based on the ongoing efforts of one institution.

WHO IS CORESTATES:

CoreStates Financial Corporation is currently the 36th largest bank holding company in the United States when measured in assets. However, we are in the top ten of the largest Visa issuers, international funds transfer banks, and check processors. We are the proprietors of MAC which is either the first, second, or third largest shared ATM network in the country, depending on which measure you prefer. We process over a million IMS transactions a day through three IMS systems. In addition, we do approximtely 250,000 TSO transactions each day and 100,000 CICS transactions a day. Our internal networks connect over 8000 devices spread over 4 continents to our Philadelphia systems center. The majority of the 8000 devices are located within a 150 mile radius of Philadelphia.

WHAT IS CAPACITY PLANNING:

It used to be that we could plan for increased capacity in our mainframes by adding a processor when we reached between 70 and 90 percent of the capacity in the system. Over the last several years, however, our mainframe applications have had very rapid, very large growth which has lead to an occasional short lead time to address a capacity issue. If we begin to exceed our capacity constraints, the result is that established businesses suffer increased response times or missed deadlines due to new businesses being brought on or existing businesses growing. We had such a situation at CoreStates about three and a half years ago. At that time we determined the way to stop this from happening was to split the responsibility for performance and availability. By availability I mean, "Is the machine up and running?" By performance I mean, "How fast is the machine running?" We not only must give our users dependable service, but we must also give them a level of service which allows them to be effective at their jobs.

PERFORMANCE VS AVAILABILITY:

In the three years since implementing this split in responsibilities, we have noted several things. We know that the people responsible for availability (technical services and data communications groups) usually know exactly what to do to make the machine run fast. They ordinarily have a good idea where the bottlenecks are and they know what they have to do to solve the bottlenecks. However, the availability people have two problems. First, they don't have the time. They are buried by the burden of day-to-day operations and the press of production problems. Other priorities are often seen as more important that capacity planning. Second, they typically have no inclination to do the detailed data collection and data analysis that is required to study the problem.

Performance management at CoreStates consists of two fundamental activities. First, we create "wizards". "Wizards" know their area of expertise intimately and are not tasked with operational responsibilities. A "wizard" can recognize small disturbances in detailed data and knows the implications of these disturbances. By identifying abnormalities at this level, big problems are not allowed to develop without warning. "Wizards" are created implicitly through the data collection and reduction activities. By writing the data collection programs, maintaining these programs, and building reports, the person responsible becomes a "wizard" in that area.

In addition, performance management means developing a partnership with those people responsible for availability — the technical services people or the data communications group. As performance analysts, we are enjoined from making any changes in the machines or networks directly, since any change could impact availability. Thus if we wish to see our recommendations manifested in the world, we must convince the availability people that our recommendations need to be implemented. And the side effect of this is a viable system of checks and balances.

This way of doing performance management works well. In the last three years, we have quadrupled our total transaction load (400% increase), cut our response time by 70%, and only increased our hardware by something less than 200%. So with substantially less hardware per transaction, our response time has actually decreased. We have also avoided crises in capacity.

APPLICATION TO NETWORKS:

Networks are critical to the delivery of our services, whether it be ATM's, home banking, funds transfers or any other thing our users might be doing. The way the users measure their service is by the time it takes for them to get their response back. The networks are the major component of the response time. In some cases, they are the overwhelming component of response time. If we squeeze another tenth of a second out of our IMS response time by tuning DASD, our users would not notice. We will have accomplished nothing for the user. But, if we cut a 4 second communication delay in half, the user would notice.

Due to growth in new and existing business over the past two years, we have doubled the number of networked terminals (from 4000 to 8000 devices). As we installed networks to meet these additional requirements, we had to keep the initial networks in place to meet established business needs. Typically, there wasn't time for a planned consolidation or the protocols were incompatible or expenses were perceived as too great. Due to the extremely rapid and large growth in attached devices, we have networks which overlap and contain architectural paradoxes. And it is these networks that are the largest contributor in most cases to how our users perceive our level of service to them.

Our networks are in the same situation that our mainframes were in several years ago during our more severe capacity situations. The networks are very complex and have grown very rapidly in a short period of time. The people who are keeping the networks running know how to make the networks run and run fast. These people, however, have neither the time nor the inclination to do performance engineering. A separation of responsibility for performance and availability is required.

We believe performance management of the networks is critical to our business goals. Past experience dictates that the key to network performance management is the spawning of a wizard through detailed collection and analysis of data and the forming of a partnership with the people who are tasked with network availability.

NETWORK PERFORMANCE:

We see that there are fundamentally two components to network performance management and capacity planning. The first component is clearly throughput, which is a function of workload and topology. Methods to increase thoughput are well known and in fact, we've had some successes in this area already. The second, and perhaps less obvious, component of network performance management is cost performance.

It is in this costing regard that networks are significantly different from mainframes. Mainframe cost performance is a function of capital costs. We would like to defer or skip altogether buying another mainframe as our workload grows. Expenses are typically very small in relation to the capital cost. The only major recurring cost is depreciation, which has really already been paid. With networks, on the other hand, the initial capital cost is quite low with regard to the monthly recurring costs, typically on the order of 3-4 months of recurring costs. It is this recurring cost, the out-the-door money each month, that we wish to minimize in the network environment.

Our approach at CoreStates is to look at the network throughput and the network recurring costs, both separately and together.

In the throughput analyses, it is useful to look at factors which affect terminal response time in detail. These factors include terminal workload by application, inbound and outbound message size, peak workloads, line characteristics, line speed and utilization, and the physical layout of the network such as the number of drops per line. In cost analyses, it is necessary to examine in detail the tariff for each drop and compare the costs per terminal within network, by area, and across all networks.

SUCCESSES:

Our first success in throughput analysis was in our Bank of Delaware operations. When we looked at the sustained transaction rate, the linespeed and the average message size, we found that the response times that our line monitors were reporting were not possible. Given the amount of work that had to flow through the lines, the people had to be waiting longer. When we investigated, we discovered queuing at the control unit. After the users hit the enter key, they were waiting some length of time before the message disappeared from the screen. The technical services group installed a super-optimizer to further compress both inbound and outbound messages and the data communications group changed the line characteristics to full duplex. When this bottleneck was eliminated, the response time dropped about 50% and there was an immediate increase in throughput by about 20%. We anticipate this will have a direct impact on revenue because these transactions are being generated by our Visa collectors.

We have also had some success in reducing network recurring costs. First, we must recognize that the tariffs in this country for leased line service are very complex. Now, in every jurisdiction, there are anywhere from three to five ways, depending on how clever you wish to be, to price leased line service. Since tariffs are so variable and disagree with each other so much, that it has proved worthwhile to have someone who understands these tariffs in detail in those geographical areas in which we have active networks. On the first network that we looked at, we examined the tariffs in detail and we were able to reduce the cost of the network monthly recurring costs sharply. We found that the tariff that had been installed was not the tariff that we had designed. The reduction in leased line costs effected by this detailed examination was on the order of 20%.

CONCLUSIONS:

Capacity planning is not a function done for the benefit of the capacity planner nor for the benefit of technical services group or data communications group. Capacity planning fundamentally must be a function done for the benefit of the business. The impact of good capacity planning is to generate more revenue or lessen expenses while allowing the same or greater level of service to our customers.

It is the partnership between the availability and performance teams which allows for effective capacity planning. In this environment, the capacity planning group looks good only when the people who use the results of capacity planning look good. When the technical services group and the data communications group are able to be more effective at their responsibilities because of the work we have done, then we have been effective as capacity planners and users benefit.

SOME EXPERIENCES OF DATA CENTRE RELOCATION

Brian Day
Harwell Computing Centre
Harwell Laboratory

ABSTRACT

Relocating part or all of a Data Centre can be a traumatic experience; and if the move is to be successful, it requires careful planning and implementation by a team of skilled people.

This paper will focus on two actual cases:

* the migration of work to a new computer system running a new operating system;

* splitting an existing Data Centre, with the consequent migration of a large portion of the workload.

The issues discussed, both technical and management, will range from site planning through user/data migration to a new production environment. The specific experiences will be used to illustrate more general principles.

INTRODUCTION

Change is a fact of life in the computing industry, and there can be few installations which have no upgrades - either hardware or software - during the course of a year. In fact many "simple" changes will most likely be routine scheduled events tracked by the standard change control mechanisms in place at the site.

However, the relocation of work, either across dissimilar processors or from one site to another, can result in extremely complex problems; especially if continuity of service is a paramount requirement. This requires a high level of project management, extremely careful planning and implementation by a team of skilled people.

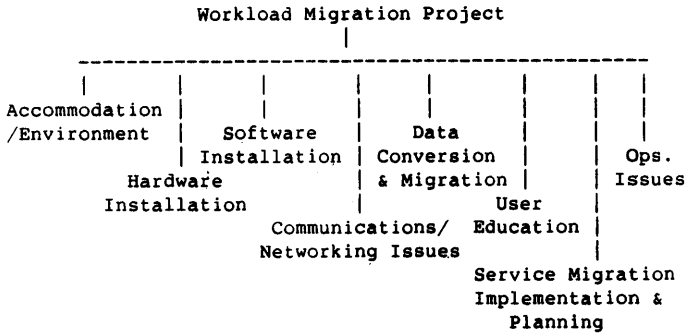
The remainder of this paper will focus on the planning and implementation phases of workload relocation, and will be illustrated by two specific examples.

RELOCATION STRATEGY AND PROJECT PLANNING

As mentioned above, careful planning is of the essence; and some form of project control methodology is vital. However, this does not automatically mean a computer based system. In the planning phase the computer can be an aid to successful project management - but it is not an alternative to really understanding the problems, serious thought and careful planning.

In principle, the project manager should focus on the goal he wants to achieve, and to establish the route to the goal; not (yet) think in terms of how it will be achieved. This definition should produce a set of milestones, which if well chosen will be independent of the method of achievement. For example, with reference to data centre relocation, a milestone might be: "when the user data has been migrated"; however the method of migration need only be determined when earlier tasks are completed and a better picture of further requirements is known. In this way a plan can be built through a hierarchy of milestones with objectives planned and achieved with a minimum of revision of implementation details.

Viewed from another perspective, the project will probably break down into a number of inter-related sub-projects which will have to co-operate in achieving the milestones referred to above. With particular reference to workload relocation, key sub-projects might be defined as follows:



Each sub-area could have a responsible project manager whose job it would be to produce detailed sub-project milestones, activity plans and resource schedules. These can then be integrated to form a total plan.

Obviously this is an extremely complex subject, and one cannot hope to cover it in great depth in a short paper. In addition it is unlikely that any two relocation projects will be the same, being very dependent on site specific detail. However, to illustrate some of the issues involved, the remainder of this paper will briefly discuss two projects the author has been involved in. Specifics of these migrations will be used to demonstrate more generally applicable concepts.

PROJECT ONE: CRAY-1/COS TO CRAY-2/UNICOS MIGRATION

This project involved the installation/upgrade to a CRAY-2 "supercomputer", and the migration of a production workload from the batch operating environment (COS) on the CRAY-1 to an interactive UNIX† environment (UNICOS) on the CRAY-2. Thus in this case not only was the underlying system and management philosophy different, but also the user interface was completely changed requiring a significant level of re-education.

† UNIX is a registered trademark of AT&T.

The education issues were therefore twofold:

* educate and convert the systems support personnel

* educate the end-users.

In our case, the problem was compounded in that there was little or no UNIX experience in house; and since the CRAY-2/UNICOS system was just out of development, the manufacturers documentation was still extremely sparse. There was a high degree of re-orientation requiered by MVS specialists; however, eventually the support personnel became familiar with the new system and even liked some of the features. Education courses were arranged for the end-user, who was often quite receptive. However, our experience shows that one should not underestimate the time taken for education and that since many users treat the computer as a tool, they are not necessarily interested in the "bells and whistles" extolled about the new system. In fact the user population polarised into two camps:

* Those who convered immediately

* and those requiring 'encouragement', and essentially left migration until the last minute

Our other major lesson was that in installing very new systems one has to be prepared to "make do" with less than a complete system. This can cause considerable stress, both for the user, and the support staff. However, on the other hand, at this stage the manufacturer is willing to invest large amounts of effort in you. This means you have a high level of input into how the embryo system develops, and you have direct contact with the developers.

Data migration was another issue during this conversion. Not only the "file-structure" as perceived by the user changed, but even the internal philosophy of handling files changed. This makes data migration a non-trival exercise. Again, there was a heavy educational requirement. However, because we had the luxury of running the old and new systems in parallel, both of which were accessible via the same front ends, the burden of conversion was done by migrating the files to the front-end processor and re-acquiring them onto the new system. Thus a large amount of the migration was left to the user himself to take at his own pace, although there were a few "carrots and sticks" to help the migration along.

PROJECT TWO: WORKLOAD MIGRATION TO A NEW MVS/XA DATA CENTRE

This project, in many respects, was the more "usual" in that it involved the relocation of a significant user population from an existing MVS/XA environment to a new MVS/XA data centre. This involved the construction of a new machine room complex, the installation of the hardware and software, data migration, and a "sudden death" cut-over which was to be transparent to the end-user.

Firstly, consider the environmental and construction phases. The complexity of this phase is very dependent on the size and location of the building. Some may be lucky enough to have a "green field" site; or, as in our case, we were in the position of constructing a machine room within an existing aircraft hanger! This hanger was about 400 feet from our existing room, and we were to convert one corner into a functional machine room to accept a IBM 3090-200E within 135 days. As usual we encountered the normal problems in clearing the site; however we also had to be cognisant of experimental facilities operating in the rest of the hanger. This posed problems because of their affect on us; however we also had to be aware of the effects of our construction work on them. These experimental facilities included a nuclear reactor(!), and an experimental rig capable of producing high levels of RF radiation. Thus before building could start, radiation surveys - both nuclear and RF - had to be undertaken. Following this, safety walls were erected to give a nuclear radiation free area, and the construction of the machine room had to be modified to incorporate shielding into the walls to produce a Faraday cage to shield the computer from the RF radiation. Other issues in the construction phase mainly revolved around delivery schedules, and the general requirement to co-ordinate the sub-contractors.

Our experience shows that no problems are insurmountable, and after accepting these challenges we now have a fine new machine room, which was constructed within the prescribed period - much to the credit of the contractors involved.

The other main issue of this conversion was the mechanism of data migration. The user group being transferred consisted of about 300 scientists and engineers working on the JET Joint Undertaking at a sister site 10 miles away. The experiment produces large quantities (>400 Mbytes per operational day) of data which was previously stored and processed on an IBM 3084Q. The migration to the 3090 would involve the copying of this raw-read only data, as well as migrating the large amounts of processed and analysed read/write data. For a number of different reasons, the target 3090 machine was to be located at Harwell but would not share DASD in the final configuration. In addition, the machine room separation precluded the sharing of an MSS which housed much of the experimental data. Thus how should migration be achieved?

Obviously the read-only data can be copied at any time, and is in many ways the easiest to handle. It was copied into a second MSS and onto 3480 tape cartridges attached to the 3090 system. This process also allowed the data to be re-validated. However, how could the "user data" and processed data which was updated in a volatile TSO environment be migrated, while maintaining a production service throughout. A number of senarios were discussed, but the two serious contenders were:

* to transfer the active user data over a "week-end" with the users and set up some mechanism to automate the transfer of the rest of the data on request;

* to share some DASD between the 3084 and 3090 on an interim basis, so that data could be moved to the shared DASD, and after user relocation the access would be from the new CPU.

Naturally the second senario is conceptually the simplest, and was the one chosen. We were extremely fortunate in the proximity of the two rooms!! In addition we had sufficient space in the 3084 computer room to allow the early installation of some of the 3090 DASD. Hence preparation for the migration was started prior to the 3090 delivery.

The alternative senario would probably have worked if the rooms were separated by a longer distance, however experiments over a CTC link between a 3084 and 4381 showed that the data throughput rate was significantly less (100Kbytes/sec) than the theoretical speed of the link. In addition, the CPU consumption during the transfer was rather prohibitive.

Using existing 'backup' communication lines, and with the aid of some 'borrowed' kit; it was possible to set up parallel communications links between JET and the 3084 and 3090 systems. Thus in the end the "sudden death" cut-over, which had been dreaded was rather an anti-climax, as it was possible to switch between the systems, and switch back should major problems have

been encountered.

In practice, the cut-over went extremely smoothly, and the project completed ahead of schedule. This was due in no small part to the extremely close collaboration between Harwell systems support, IBM, and JET data management staff. Without this close co-operation, success would have been impossible.

CONCLUSIONS

The examples above show that there is an extremely broad spectrum of requirements in relocating large portions of a workload. The variations on the requirements are only matched by the variation in the solutions. In the CRAY conversion the user migration was phased, and much of the migration was user initiated using information and tools supplied by the data centre. In the JET case, the migration had to be 'en mass' because of data sharing requirements and were done by systems and JET staff. However, in both these dissimilar cases, acceptable solutions were found to the problems.

I hope this paper has shown that workload relocation can be exciting and challenging, and that with proper planning and involvement of skilled people, the probablility of success is high and the risk can be minimised.

ACKNOWLEDGEMENTS

I should like to acknowledge the work and effort of all the people involved in the projects referred to above; not only those in the Harwell team, but also to CRAY Research, IBM, and JET staff who collaborated on these projects. In addition I should like to thank Harwell and JET for permission to discuss their migration experiences.

THE ROLE OF CAPACITY PLANNING IN CONSOLIDATIONS, ACQUISITIONS AND MERGERS

Holly E.K. Verbil
Bank of New York

ABSTRACT

Economics, which reflect the business environment also effect the requirements for the service needs of any organization. In the current business climate, mergers, consolidations and buyout situations are becoming a common mode for businesses to increase their assets. As Data Processing is generally considered a liability to the organization, the burden of proving the viability of any consolidation should be on the Data Center (because of its service orientation), specifically, Capacity Management. This paper will discuss the life-cycle of the consolidation/merger plan with emphasis on the role of Capacity Management. Topics include: reporting requirements on key workload indicators (before, during and after the consolidation/merger), characterization of applications (historic profiles), resource requirements (identification of hardware and software requirements necessary before, during and after the consolidation/merger), the use of modeling throughout the process, critical paths in workload movements, and budgeting properly for reserve capacity after the move/consolidation.

INTRODUCTION

The Bank of New York is currently going through a Data Center consolidation effort. During the same time period, we also followed through with the acquisition of another bank. As many businesses are finding, ventures such as mergers, acquisitions and consolidations are financially economic ways to expand business. This paper reflects the internal efforts of Data Processing on behalf of the management decision to consolidate or merge in order to increase business assets.

HISTORY

The Bank of New York, through years of growth, political, and economic changes ended 1986 with 4 multi-user Data Centers, all within a 50 mile radius of New York City. Several (14) single user mini computer satellite operations also fell under this umbrella. Though an interviewing process found that independent users felt that this decentralized distributed Data Processing environment was an acceptable situation even one to be envied, the majority of users were frustrated by the inconsistency of the processing services they were receiving. For senior management, Data Processing has always been a necessary evil, and a reduction in bottom line cost of services to support the Banking Operations was an intriguing idea for them to consider.

Financially, a plan to consolidate the Data Processing function, and coordinate hardware and software by product lines offered a cost effective method for providing efficient and effective service. Thus was born the consolidation.

Clearly, much time and research went into the consolidation decision, but the idea was left to be planned, organized, and carried out by the Data Processing Division of the Bank. The initial group of D.P. managers (representatives from the four major centers) were presented with the challenge of consolidation along with a concise breakdown of certain financial conditions. Facts included; cost per square foot of Data Center space in each of the four locations, an overview of the processing effort within each Data Center, the processing power (MIPS) for each, and the last year's budget figures for each Data Center. With these facts, they were to form a plan. In the following session, they devised an overall strategy which states the objectives of the consolidations.

The strategy established four major goals;

- Improve service quality: It was felt that the structure of Data Processing hindered the quality of service to its users.Specifically, the strategy established the requirements for the newer standard environment software common to all Data Centers, improved controls and management disciplines that focus on both accountability and responsibility, and a reduction in the complexity associated with controlling more than one major multi-user Data Center.

- Reduce operating costs: Redundant positions, inefficient usage of hardware and software resources, and the ever-rising expense of office space in New York City are some of the areas where costs could be reduced. Specifically, the strategy identified net personnel reductions, net reductions to hardware/software and improved facilities and telecommunications.

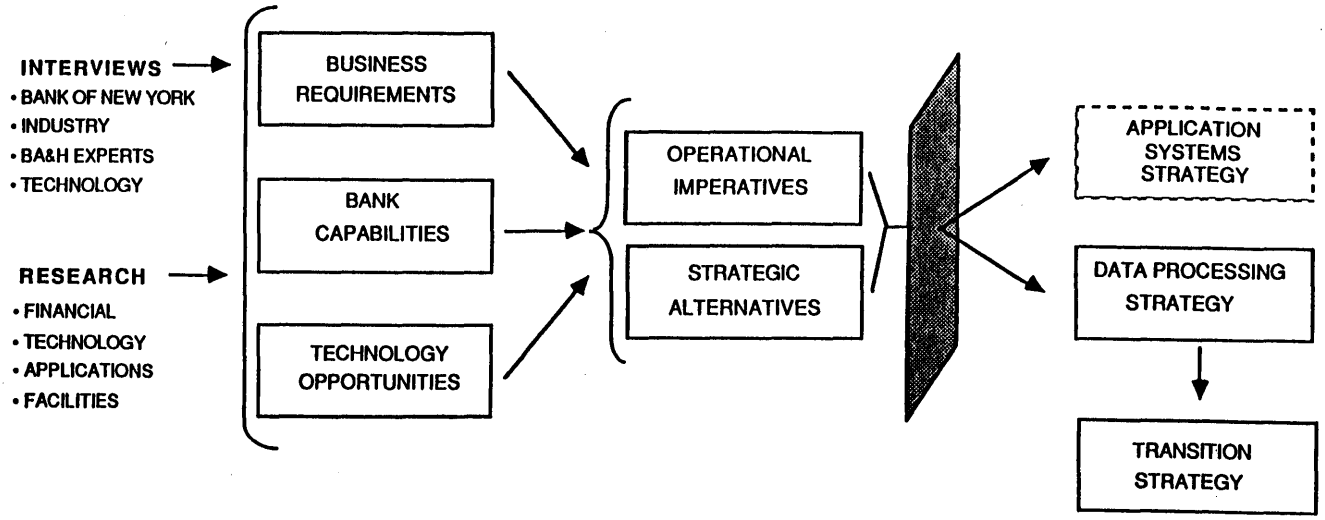
- Increased information availability: Inconsistencies in software and hardware options available reduced the effectiveness of the applications programming department and was severely limiting the ability of users to access critical business information. Specifically, the strategy establishes a common environment, a common set of standards and tools, and the development of a focal point to ensure that satellite user computing is effectively used.

- Improved Product Innovation: Barriers built up through years of unreliability and inflexibility prevented the Bank from taking advantage of newer and more innovative technology. Specifically, the strategy removes these barriers and provides a more flexible standard environment.

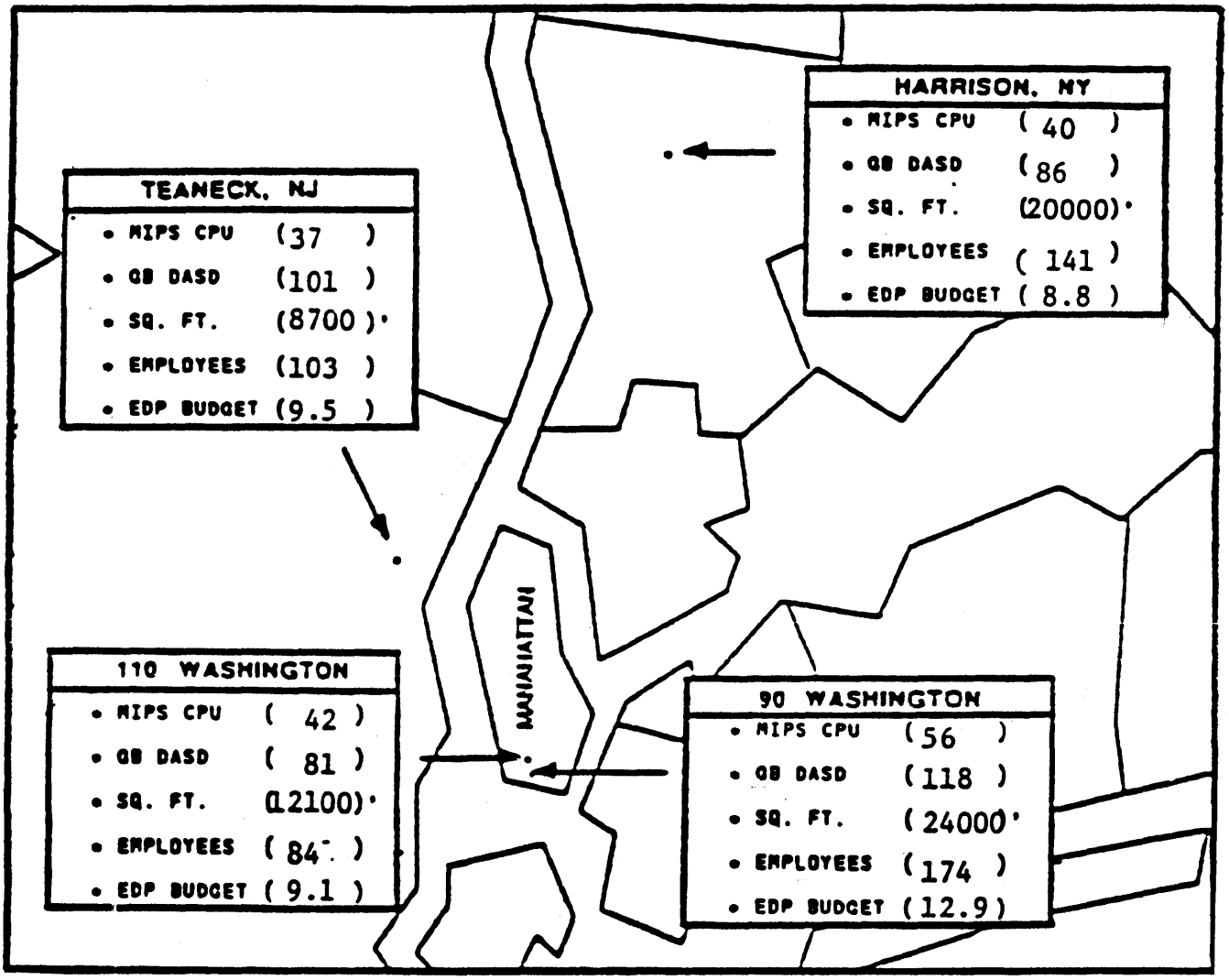
The initial strategy called for a phased approach to the consolidation. The first phase involved the consolidation from four to three major Data Centers. Financially and business wise the redistribution of one of the New York City Data Centers made the most sense. Using this as our direction, we proceeded to divide the applications (on a global scale) by product line back through two of the remaining Data Centers. Our motivation for moving applications to specific locations carried forward that idea. All of the Trust/Security processing related applications were to move to one location and the Bank Operations and General Business applications were to move to another location. The other New York City Data Center was left untouched by this phase of the consolidation for future strategic plans.

APPROACH...

THE STUDY FOCUSED ON IDENTIFYING BUSINESS DRIVEN IMPERATIVES AND ON DEVELOPING APPROPRIATE DATA PROCESSING AND TRANSITION STRATEGIES TO SUPPORT THESE IMPERATIVES...



APPLICATIONS SYSTEMS, WHICH ALSO ARE AN INTEGRAL ELEMENT OF HOW INFORMATION TECHNOLOGY CAN SUPPORT ACHIEVEMENT OF BUSINESS OBJECTIVES, WERE OUTSIDE THE SCOPE OF THIS STUDY...


HARRISON, NY
• MIPS CPU (40)
• GB DASD (86)
• SQ. FT. (20000)
• EMPLOYEES (141)
• EDP BUDGET (8.8)
TEANECK, NJ
• MIPS CPU (37)
• GB DASD (101)
• SQ. FT. (8700)
• EMPLOYEES (103)
• EDP BUDGET (9.5)
MANHATTAN
110 WASHINGTON
• MIPS CPU (42)
• GB DASD (81)
• SQ. FT. (12100)
• EMPLOYEES (84)
• EDP BUDGET (9.1)
90 WASHINGTON
• MIPS CPU (56)
• GB DASD (118)
• SQ. FT. (24000
• EMPLOYEES (174)
• EDP BUDGET (12.9)

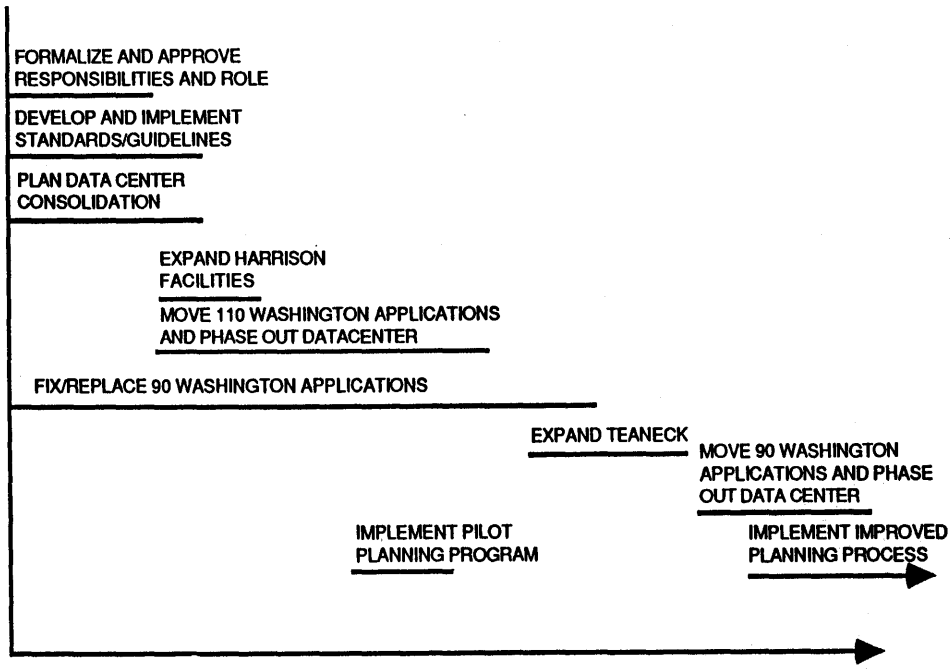
As the significance of this project began to sink in throughout the bank's D.P. community, we launched into implementing the plan. Everyone realized at once, there were to many unanswered questions, too many unknowns. This brought us to the first step in the plan, "data gathering". We identified several things we needed to know about each of the Data Centers before we could do much more work. Our checklists called for lists of software and releases, hardware inventories, major workload characteristics and some basic CPU utilization statistics. Other things such as lists of applications, number of jobs per day, week, and month, number of tape amounts, type of job scheduling, library structure, type of cataloging, JCL handling, data security procedures and naming conventions were also identified. With most of these things gathered from unknown sources (each Data Center manager brought back his own information), validation of some of the information was required. This task was performed by Capacity Management, where there was a general request to gather SMF data from all of the Data Centers.

Organizationally, Capacity Management reports through each of the Data Centers to independent Data Center Directors who report to a common MIS Director. Their continuity is managed solely by their efforts to remain in sync with each other. The consolidation brought with it the need to manage a combined resource database through which general information and special requests could be handled in a standard way. All of the individuals involved with Capacity Management for the bank agreed that a central repository for our resource data (SMF, RMF, CMF, and IMS) was necessary in order to manage the large amount of data accumulated. Through the data, CPU utilization statistics, major workload characteristics and profiles, and job statistics were tracked by application. Start I/O's, tape mounts, and transaction counts were also reported. All of the Data Centers had established different levels of tuning, from the very basics to forecasting and modeling, so the idea of gathering all of the data together did not constitute any major procedural changes for the Data Centers. Capacity Management worked with the software systems group to modify the IPS (Installation Performance Specifications) on each of the systems to select and identify report performance groups which could then be tracked to indicate any changing trends. The information most difficult to ascertain included physical inventories of both software and hardware and terminal transaction relationships. On the communications side, we needed to learn about how the Data Centers were connected, what kind of data was going across the lines, and how much of it there was, what the anticipated changes would bring, and how much of it there was, what the anticipated changes would bring, and how quickly they needed to meet the requirements of the transition moves.

A detailed plan was developed by which each "application family" would move into its new location. The approach was to follow the plan (as best it was possible), move the first family, then reevaluate and resolve any differences, and modify the plan accordingly. The assumption was that every application had similar basic characteristics, with modifications only in the quantity to which particular things applied. The plan called for checklists, task lists and schedules to be created and maintained. It called for scheduled meetings and the irrevocable commitment by the Data Centers to meet the requirements of the transition with a minimal impact on the current services available.

During the first four months of the project, Capacity Management began the data collection process. Since this was still the planning phase of the project, during weekly meeting charts and graphs depicting some of the Data Centers workload characteristics were distributed. Some statistical reports were developed to assist the D.P. managers in understanding the impact of the consolidation efforts on their particular Data Center. While the resource database continued to increase, a trending history began to

TRANSITION PROJECTS


FORMALIZE AND APPROVE RESPONSIBILITIES AND ROLE
DEVELOP AND IMPLEMENT STANDARDS/GUIDELINES
PLAN DATA CENTER CONSOLIDATION
EXPAND HARRISON FACILITIES
MOVE 110 WASHINGTON APPLICATIONS AND PHASE OUT DATACENTER
FIX/REPLACE 90 WASHINGTON APPLICATIONS
EXPAND TEANECK
MOVE 90 WASHINGTON APPLICATIONS AND PHASE OUT DATA CENTER
IMPLEMENT PILOT PLANNING PROGRAM
IMPLEMENT IMPROVED PLANNING PROCESS

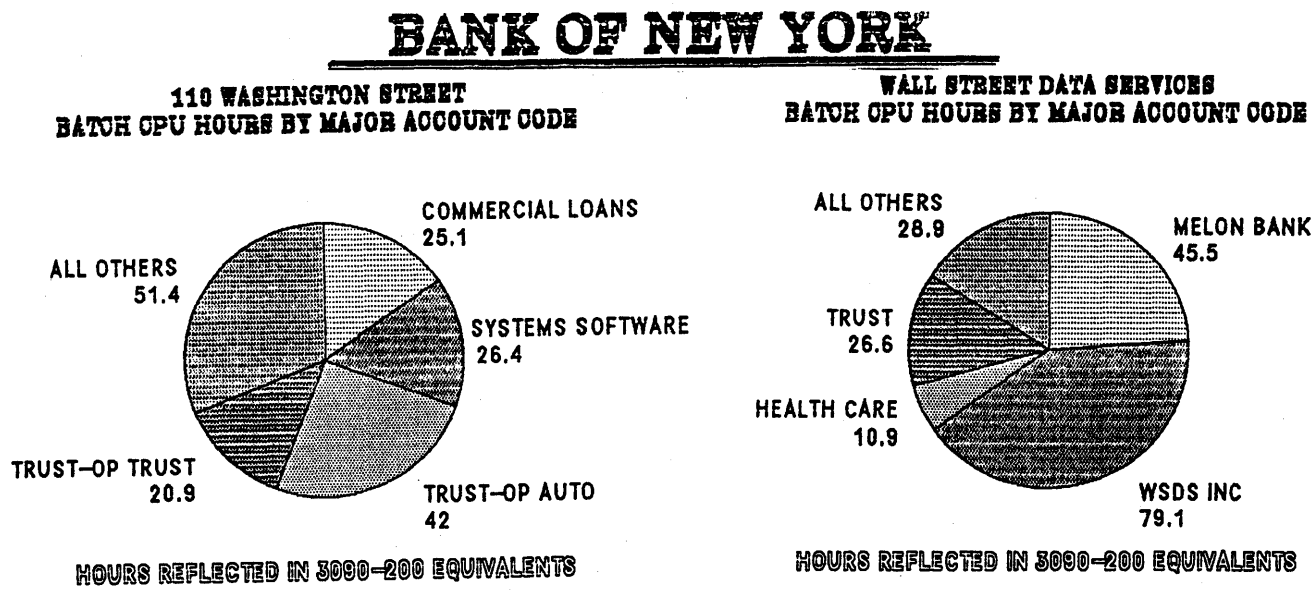
develop for each of the families. This information would later be used to assess the impact of each of the moves. Capacity Management's expertise was also required for the next step of the project. Profiles identifying all of the resources for each of the applications needed to be defined. Interviews with users, application programmers, and group leaders were an important step in the background history included in the profiles. As the history accumulated, profiles were built and validated to insure accuracy. We felt that by developing a solid historical base for each of the application families, we would be best prepared to move an application family to its new location.

A major campaign to inform the other divisions of the bank about the consolidation was developed. This was a twofold mission, to enlist support from them in regards to problems with logistics, and additional procedures required and notify them of the transition changes and what impact it would have on each of their divisions. These were high level meetings designed specifically to deal with divisional concerns. An orientation brochure about each of the Data Centers was distributed to introduce the divisions who had never been directly involved with the Data Centers to them. The meetings canvassed Data Center users feelings and reservations about the transition moves and reassured them that their current service would not be hampered in the move process. We also met some specific concerns head on by requesting their help with the moves and the historical data gathering. We asked them if they could anticipate any benefits from the moves, by asking them what they found particularly disturbing about the Data Processing service that they were currently receiving. Some of the responses required simple resolutions such as getting printers into user areas, having more consistency to the scheduling of jobs, and the reliability of data provided, all of which we felt confident that we could support.

It soon became obvious that the tasks were growing beyond the limits of the initial group. We began to bring the "doers" into the team. This constituted the end of the planning phase and the beginning of the move phase of the Data Center Consolidation. The new team members were the working core of the team. They were managers of the people who would actually be doing the work, they were the project leaders, and they were responsible for each application family move. I must say here that the individuals chosen were done so with great care and concern. Nothing that we planned could have prepared these people for the extensive work ahead of them. They have done an extraordinary job under incredible circumstances. The group was still limited, and while many people would work on this project throughout time, we had to maintain the day to day environment and make every effort to stabilize the incoming work and quickly fold it into the normal production flow.

As the first move loomed over us, doubts began to surface as to whether strategic planning was the best way to accommodate changes like moving an application. Some wondered if it wouldn't just be easier to move the applications and have it over with. Those of us who were used to planning, had to allow for these frustrations, knowing that eventually the planning would prove to be the survival of an undertaking such as this.

Schedules got tighter, time got short and people got incredibly busy for the next few weeks. The weekend prior to the first move (scheduled for the following Monday), all of the checklists were checked, tasks completed, and schedules updated for a smooth start up. People were in place and ready to respond to any situation as Monday arrived, and the system went live. The batch cycle went relatively smoothly, with the exception of some last minute JCL changes and the on-line work started on schedule. The problems that did surface were manageable and were not considered "show stoppers" to the overall objective.



As the first "debriefing" meeting (after the first move) people were critical of their work, but none the less, the task had been achieved, and to the bank we had done the job as promised. We charged into the next application "family" promising not to make the same mistakes as the first, knowing that we would make different ones instead.

By the time we began planning for the third move, things began to be routine in nature. The application "families" that were moved began to fold into the normal production workload of the Data Center.

Capacity Management remained in the middle of the process by reviewing resource data which revealed the changing characteristics of the Data Centers. Graphs, charts and statistical data were provided to help explain the impact of each of the moves in both Data Centers. Capacity Management was also responsible for reporting on and validating the previously estimated increases in the production workloads.

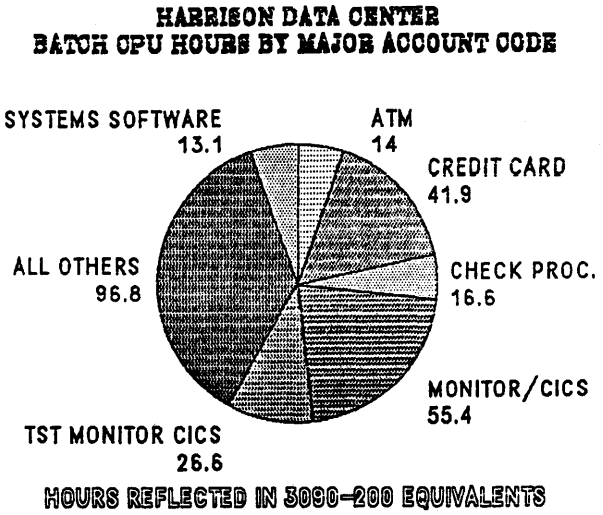
Hardware and support software had to change as well. Additional DASD was required to support the new applications in one Data Center while in another Data Center the DASD requirements dwindled. The problems with one for one movement became obvious when we started researching the methodology used in the DASD space allocation process. The space freed up by the movement of applications was not readily available because of a commitment to backup a moved application for a limited time and because the space was allocated across several volumes with a complicated structure of files and pointers which might jeopardize the remaining applications. We budgeted hardware and support software to be installed and tested at least one month prior to the particular application move they were scheduled to accompany. As equipment was freed up in one location, and was required in another location we chose to take advantage of some short term leasing options to avoid any "push-pull" situations. To compensate for some possible problems in our ability to backup the new configurations, the final Data Center structure called for some mainframe changes as well. This allows the bank to take full advantage of the consolidation to replace older, less dependable equipment with newer, more advanced technologies and position itself to respond to future needs in a timely and cost efficient way.

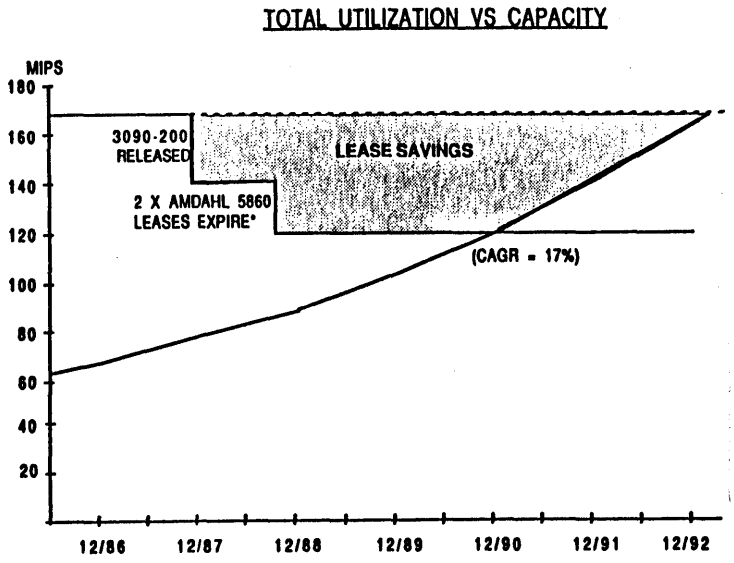
CONCLUSIONS

By involving Capacity Management in the planning phase of this project, we were able to avoid getting caught up in the process by a lot of unanswered questions, and still limit the core group. The planning tasks were made easier by our technical knowledge and our ability to translate the needs of the project into the impacts on each of the current Data Center environments. We were able to help management understand the changes that would be felt by the Data Centers that would be gaining applications by pointing out that it would affect processing windows, on-line timings and that there might be some batch conflicts that would require analysis.

There are a few points that I wish to bring out as I conclude this paper. Capacity Management can make a major contribution to the strategic planning of the business entity known as Data Processing. Our technical knowledge, combined with an awareness of the Data Processing structure gives us a good sense about what is happening within a given complex. In the case of a consolidation, merger, or acquisition, our skills and efforts on behalf of the project offer a wealth of information to management who may not realize the implications of what they are about to commit to until its too late!

As a manager of Capacity Management, your direct involvement in the processes associated with a consolidation should be invaluable to the project. As an employee working in capacity management, you can make sure that your skills and contributions are noticed by providing the right kind of data to your management team in order for them to make informed

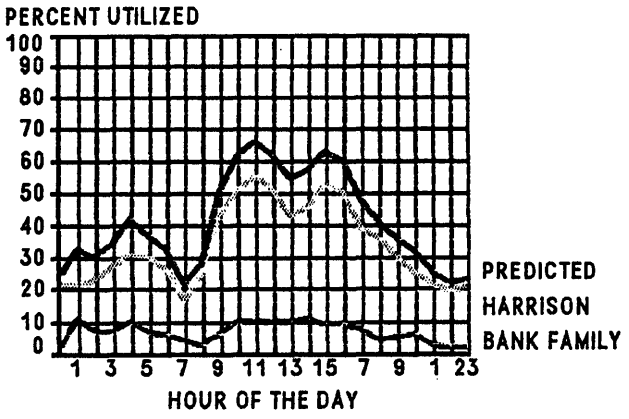


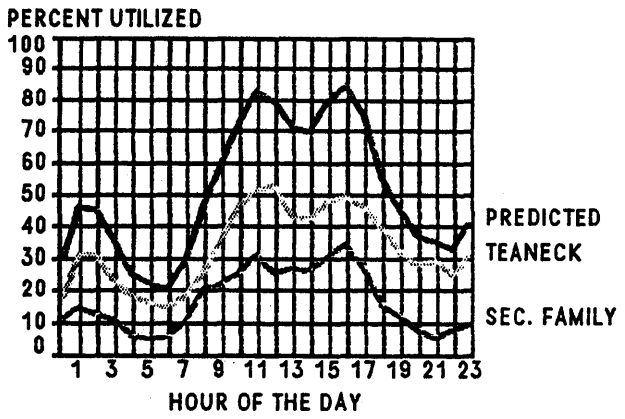


STRATEGY - ECONOMICS...

AVERAGE SAVINGS OF $2.8 MILLION PER YEAR FOR THREE YEARS STARTING IN 1988 CAN BE EXPECTED DUE TO HARDWARE CONSOLIDATION AND MORE EFFICIENT UTILIZATION...

CPU CAPACITY

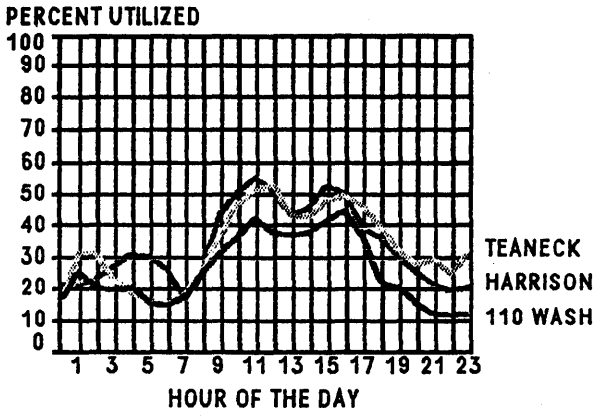




HARRISON, TEANECK & 110 WASHINGTON

NOTES :

UTILIZATION WAS DERIVED FROM AVERAGE CPU SERVICE UNITS FROM THE MONTH OF FEBRUARY EXCLUDING WEEKENDS AND HOLIDAYS.



business decisions. Capacity Management can support the decision making process by utilizing their skills to provide a statistical information base and analyzing the database for trends, abnormalities, and conflicts. Because the training (backround and education) of an experienced Capacity Management person focuses on the impact of external forces on internal Data Processing functions, analysis of the business decision can be broken down into its integral elements to provide greater understanding of any situation. The business decision can be reduced to facts on both the economic and political levels. Prudence requires a careful weighing of both sides, but the elements of decision can be reduced in such a manner as to provide senior management with the essentials required for informed decision making.

Capacity Management should expand its horizons as well. Consideration should be given to initiating such thought provoking things as consolidating Data Center resources when and where applicable managing the capacity of computer hardware, and exploring the realm of creative finance fall into line as part of the new Capacity Manager's job. Just as you help make management decisions to purchase new equipment, consideration for reducing the bottom line are becoming part of the job as well. Capacity Management has come a long way from the days of blue sky predictions. My point is - in order to achieve the level of being part of the strategic plans of your company, you must begin to be aware of, and understand the elements of business and translate your recommendations, findings, and analysis into sound business proposals.

ACKNOWLEDGMENTS

I would like to thank Charles Barilla for his expert help in the graphics presented within this paper. Also for his help in preparing the presentation of this material. My thanks also to The Bank of New York, Data Processing Division, for providing with a vehicle to challenge my knowledge and further my experiences associated with Capacity Management.

Resource data was extracted from some internal publications, "Working Paper", A Transition Strategy for The Bank of New York, "The Harrison Data Center", An Introduction, and the Weekly Operations Statistics Report.

CAN SERVICE UNITS BE USED FOR UNIFORM CHARGEBACKS?

Carl Claunch
National Advanced Systems

ABSTRACT

Many installations determine chargeback fees by using service units attributed to the task by MVS. Multiple processor installations, or those who change the processor complex, may then experience significant variations in charges computed. This paper will discuss the nature of the service units computed by MVS, the causes of variation across dissimilar complexes, and a simple method of minimizing such variation.

1.0 SERVICE UNITS

All service delivered to a user consists of access to a resource. Resources vary in importance, as well as scarcity. The conversion of resource usage into a common measure--the service unit (SU)--allows the installation to manage in a consistent manner the diverse range of services provided to a user's task during processing.

Every resource in the system has an associated rate, which specifies the worth in SUs of each use. For example, requests for Input-Output might produce one SU per EXCP, while a very fast processor complex could produce hundreds of SUs each second of CPU access.

MVS considers three resource types in accumulating SUs:

-Processor (CPU)
-I/O
-Main storage occupancy

The unit of use for these resources is the second of active CPU, the Input-Output (I/O) count, and the number of page frames per CPU second, respectively. MVS/XA recently provided the option of counting I/O resource usage, either by the traditional EXCP count, or by the number of 8.3ms intervals of channel busy. Processor seconds are recognized separately when produced in SRB mode from those seconds spent in TCB mode.

The MVS system applies a weighting factor to these SU values being accumulated. The system options, CPU, SRB, IOC, and MSO in the IEAOPTxx member of PARMLIB, are multiplied by the accumulating SUs of each resource before MVS sums all resource usage. A user can reflect the changing importance of the resources by manipulating these weighting factors. Higher values increase the effective cost of access to the associated resource.

Workloads which use a constant quantity of the resources should be assigned the same total service unit value each execution. MVS provides reasonable repeatability when the underlying resource usage is equivalent. The I/O service units tend to vary the least, while the main storage occupancy values are highly sensitive to central storage utilization and can vary widely between runs. CPU service accumulation is fairly repeatable, but can increase when system loading is high.

Because of the good repeatability of CPU, SRB and I/O service unit consumption in a system, many installations have based fees on the service consumption of the workloads. Departmental charges can be related well to usage.

Many workloads are not reported with sufficient detail to make use of service units. For example, most online transaction oriented systems do not report service units discretely. Individual use cannot be charged back using the consumed service units, and other schemes have to be employed by the data center to recover costs.

2.0 MODEL DEPENDENT CONSTANTS

The number of service units produced by a CPU second cannot be fixed. The range of processor complex speeds is large, with MVS used on systems from 370/158 size up to the largest 3090 class systems

MVS adjusts to the relative performance of the executing processor complex by use of Model Dependent Constants (MDCs). These values define for MVS the number of SUs each second of time on the model is worth. The value for each model is set in a table coded in csect IRARMCPU in the nucleus module IEAVNP10 (see figure 1). When the operating

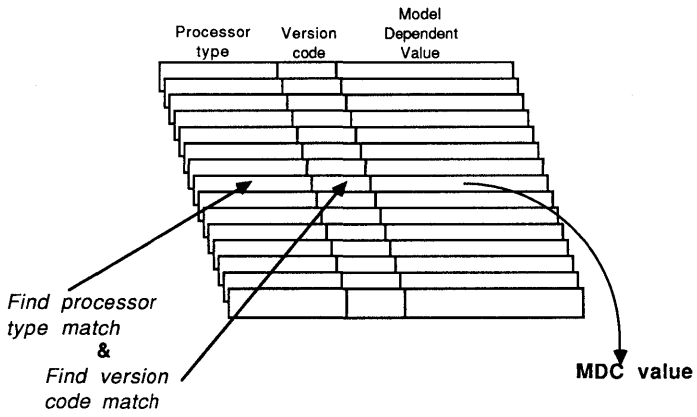


FIGURE-1

system is initialized, the CPU ID of the processor complex is extracted. This ID consists of a machine type and a version code. Combined, these values represent a specific model. The table entry corresponding to the processor type and version is used to assign the SU worth of each processor second (see figure 2).

PROCESSOR	SU/sec	Low Page Rate	High Page Rate
4381 1	122.1	37	48
4381 2	181.0	55	71
4381 3	153.9	55	71
4381 11	90.52	28	36
4381 12	181.1	55	71
4381 13	235.34	72	92
4381 14	200.04	72	92
3083 CX	166.0	43	54
3083 E	204.9	54	67
3083 EX	221.1	58	72
3083 B	306.6	80	100
3083 BX	330.7	86	108
3083 J	388.9	98	124
3083 JX	420.0	106	134
3081 D	277.1	74	93
3081 G	291.2	80	100

3081 GX	314.2	86	108
3081 K	369.4	98	124
3081 KX	399.0	106	134
3084 Q	346.1	98	124
3084 QX	373.8	106	134
3090 150	451.61	115	145
3090 150E	471.91	120	151
3090 180	750.0	191	241
3090 180E	823.53	208	262
3090 200	712.5	191	241
3090 200E	765.88	208	262
3090 300E	724.71	208	262
3090 400	667.5	191	241
3090 400E	700.0	208	262
3090 600E	625.88	208	262

FIGURE-2 MODEL DEPENDENT VALUES

A related set of constants define the page rate bounds for effective operation. These parameters default to the model dependent constants, unless explicitly provided by the user in the IEAOPTxx member of PARMLIB. These MDCs are the default for the parameters PAGERT1 and PAGERT2.

The page rate values are part of the "happy values", which are used to determine when the overall system multiprogramming level should be decreased or increased. MVS picks the high (or low) priority domain and adjusts its multiprogramming level (MPL) appropriately.

3.0 VARIATIONS IN USAGE

Four factors can affect the repeatability of service units for each workload:

-Changes in the workload itself
-System-wide conditions, such as extreme paging rates
-User changes in IEAOPTxx to the weighting of resources
-Changes in the capability of the resource, such as a faster processor complex.

Different SU totals caused by changes in the task are not an accounting difficulty. The natural expectation is that charges will vary when the work accomplished varies. No requirement is laid on system management to provide repeatable charges for such workloads.

System-wide conditions which change SU requirements tend to be most noticeable in the MSO values. Very few installations utilize the MSO service value in any significant way in the billing process. Increases in charges,when overall system load is heavy, act as an incentive toward off-peak scheduling. This paper will not provide suggestions to offset this generally positive effect.

The IEAOPTxx weighting factors should not vary across processor complexes if comparable billing is desired.

Finally, a change of processor complex that alters the workloads requirement for processor seconds can be further segmented into two effects--a change in the average processor speed which will effect all workloads similarly, and model dependent changes which are different for every workload.

Each processor type is designed with a specific set of hardware capabilities, striking a balance between performance and cost. The cost may be economic, or it may be physical, such as heat or size. The overall goals of the designer are to provide a target average performance at desired physical and economic values. Every machine type strikes its own different set of balances.

When compared to another one, machine type may be quite proficient executing floating-point arithmetic, but less so when moving blocks of data in memory. Variations in machine design result in workload sensitive variations in the rate of execution. Two models may have a single average ratio of performance, but as the workload varies, the instantaneous ratio between the models varies over a large range (see figure 3).

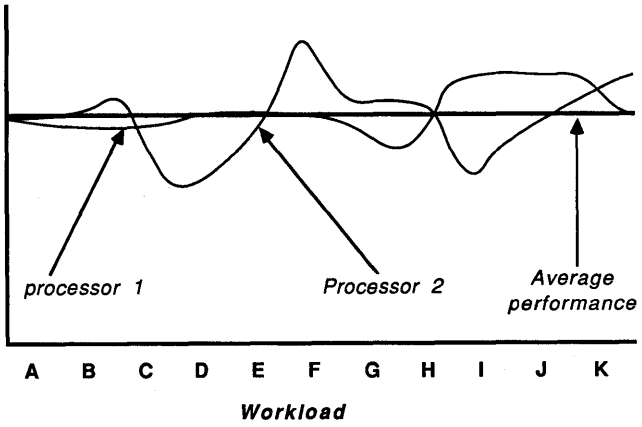


FIGURE-3

The user who upgrades a processor complex may find certain workloads benefit more or less than the overall expectation. Billing charges computed for these workloads will be bigger or smaller than before, yet the underlying work is identical.

The MDC value in IRARMCPU reflects relative performance, so that on the average, SUs accumulated for all work should be constant when the processor complex is changed. When more than one processor complex model is in use, overall SUs for equal workloads should be roughly identical. However, model dependent variations in individual workloads result in non-repeatability across models.

This effect is least noticeable when the models concerned are all of the same processor type, such as 3081 or 4381. The effect is more noticeable if factors other than cycle time comprise the difference between models, as for instance, when the amount of cache buffer is changed. The effect is significant when different processor types are involved, for example, 3081 and 3090 sharing an installation. The effect is most severe when the processor types are not closely related, for instance, with 4381 and 3090.

Since workload sensitive variations occur, the average SU rate of the MDC in IRARMCPU cannot provide repeatable SU sums for all workloads. The installation's mix of work may not average to the same relative capability as the MDC value, and individual workloads will vary by different amounts.

This paper suggests two complementary strategies to minimize the model dependent variation in repeatability. First, adjust the MDC values in IRARMCPU to produce repeatable average SUs between the models involved. Then, identify those workloads which differ due to machine differences, and provide a mechanism to adjust them in the billing process.

4.0 ALTERING MVS VALUES

Figure 4 indicates the manner in which the IRARMCPU value is used to produce a service unit rate. The values are stored in a table with an entry for each model supported by the software release, as well as spare entries

Prepare a change to csect IRARMCPU of module IEAVNP10 in the dataset NUCLEUS. Entries are doublewords; the first two bytes contain the processor type in hex. The next byte contains the version code, and the remainder is the MDC value in hex.

For example, a 3081 processor entry contains:
3081 xx00 0001 9000

An entry for a 3083 contains:
3083 xx00 0001 FC00

SU per second = 43,008,000/ TABLE VALUE

OR

Table value = 43,008,000 / SU per second

Example - The 3083JX has a service unit value of 420.0 which produces a table value of 102,400 decimal.

This corresponds to a hexadecimal value of x'19000'

Example - The table value for a 3083BX is x'1FC00', which is 130,051. This value produces a SU/sec rate of 330.7.

FIGURE-4

The table contains at least one spare entry, which is marked with a FFFF processor type.

5.0 ACHIEVING UNIFORMITY

Select from the installation workloads a set of tasks which are highly repeatable on the same processor model. Mix these tasks in a reasonable approximation of the overall workload mix. The more tasks included, the higher the accuracy.

This set of tasks becomes a baseline with which we adjust additional processor models so the accumulated service units are as close as possible. Calculate the weighted average and deviation. Use as many tasks and observations as practical to increase accuracy. The deviation percentage will reflect the change desired in the IRARMCPU constant service units per second. Adjust this value and monitor the effect. A final cross-check is the overall billing produced in production use, which should be similar when our baseline tasks have produced nearly identical results.

When average billings are repeatable, individual workload variations still exist. Cases where the variation is high, the user is concerned, and the effort is justified, merit a group adjustment. Associate workloads with similar characteristics to a grouping, perhaps by accounting code or job class. All such grouped tasks should have a similar deviation amount (and sign) from the average billing on other models.

Institute an adjustment calculation in the billing process, which applies a model dependent coefficient to the SUs. The model dependent coefficient is determined by the deviation on each model. The billing process would select a model dependent adjustment by processor complex executing the work.

6.0 SUMMARY

The system manager constructs and uses a baseline set of work to adjust IRARMCPU MDCs until average billings are nearly identical. Then the system manager measures the model dependent deviation for selected individual tasks and uses it to adjust the billing process for uniform billing across models.

A USER-FRIENDLY AUTOMATIC ANALYTIC APPROACH TO BUDGET-CONSTRAINED SYSTEM UPGRADE

Samuel T. Chanson & Angela S.O. Wong

Department of Computer Science,
University of British Columbia.
Vancouver, B.C., Canada V6T 1W5

ABSTRACT

This paper describes the development of a software package to help make computer system upgrade decisions. The package, named OPTIMAL, solves the following problem: given an existing computer system and its workload, a budget, and the costs and descriptions of relevant devices in the market, what is the most cost-effective way of upgrading and tuning the system to produce optimal system throughput? To enhance the practicality of OPTIMAL, the research followed two criteria: i) input required by OPTIMAL must be those system and workload characteristics directly measurable from the system under consideration, and ii) other than gathering and inputting the required data, the package must be completely automated and must not require any specialized knowledge in systems performance evaluation for the user to interpret the results. The output of OPTIMAL consists of the optimal system throughput under the budget constraint, the workload and system configuration (or upgrade strategy), and the cost of the upgrade. Various optimization techniques, including saturation analysis and fine tuning, have been applied to enhance the performance of OPTIMAL.

1 INTRODUCTION

As the computer plays an increasingly dominant role in today's world, a great deal of human and economic effort is being spent on planning computer system configurations to meet the service requirements of a given workload. This process of planning system configurations is commonly known as *capacity planning*; it is a process of continually monitoring a system and its workload, making projections and recommending upgrade decisions to meet performance requirements for current and future needs.

The notion of applying artificial intelligence (specifically, expert system) technology to capacity planning to increase the productivity of performance analysts has aroused much interest recently [9] [11] [15]. In particular, an expert system which uses facts and heuristics on a domain of expertise to solve a certain problem provides a powerful tool for system performance analysis. Previously, most works have been designed for the needs of particular computer systems and utilize tools specific to these systems, e.g., [10] [5] [9] [15] [16], or consider only particular units in a computer system, such as the CPU or the disk storage system [2] [7] [14]. Even for Sevcik's work [17] which considers alternative configurations to replace a system, the study was directed toward a particular case and thus is limited in its application. Furthermore, gathering the required input data and interpreting the output results in most of the suggested approaches require experience and knowledge in computer systems performance evaluation. In the realm of applying expert systems to capacity planning, the problem that we propose to solve for a given computer system is: given a budget, the current system configuration and workload, available devices on the market and their associated costs, what is the most cost-effective way of upgrading and tuning the system such that the optimal system throughput is obtained? This is the system upgrade problem that computing centre managers face from time to time when their systems saturate or approach saturation.

To enhance the practicality of the solution, two criteria are followed. The system under consideration (SUC) is a system already in existence, and the parameters characterizing the system configuration and workload are directly measurable. Furthermore, unlike most currently available capacity planning systems, after the required input data have been gathered and submitted, the solution process is fully automated and no specialized knowledge in performance evaluation is required to interpret the final results.

For the purpose of being able to apply the solution to any system in general, and the fact that analytic models have proven to be able to accurately and cost-effectively reflect the performance measures of a system [3], we have opted

for analytic models over simulation. Specifically, exact Mean Value Analysis (MVA) [13] [8] is used to evaluate system performance. Also, rather than aiming at an absolute highest throughput possible within the budget, we define optimal throughput to be the highest throughput per dollar spent. We also allow a minimum disk capacity to be specified which must be satisfied in the recommended upgrade solution. Furthermore, in almost all systems, depending on the availability of memory, there exists a limit on the number of processes that can be active simultaneously; such a limit is known as *memory constraint* which places an upper bound on the extent to which processing resources can be used concurrently, and thus on the system throughput [8]. Thus, for systems with paging and memory constraint, we further optimize the system throughput with the optimal multiprogramming level with respect to the SUC.

The remainder of the paper describes a software package called OPTIMAL that satisfies the above objectives. Section 2 describes the model and the assumptions used. Section 3 discusses some of the implementation issues. Section 4 discusses the performance of OPTIMAL. Section 5 concludes the paper. A sample session with OPTIMAL is given in the appendix.

2 DESCRIPTION OF OPTIMAL

2.1 ASSUMPTIONS

For simplicity, the SUC is assumed to be representable as a centralized single class closed queuing system model with batch or terminal workload. The possibility of extending the model to include more complex systems such as multiple job class systems will be discussed in Sections 3.1 and 5.

Placement of files on devices with the objective of balancing the load across I/O devices [1] can often improve system performance [8]. Thus tuning, with the above objective, is considered whenever it is possible to move files from one I/O device to another. It is also assumed that such tuning measures do not incur additional monetary costs.

For systems with memory constraint, it is assumed that there exist two system-dependent parameters: the maximum memory size and the maximum degree of multiprogramming. Homogeneity assumptions about the workload are also made, as described below. For swapping systems, it is assumed that with respect to the configuration and workload of the SUC, the mean degree of multiprogramming is measurable and the users share main memory equally. Thus the mean size of memory allocated per job is calculated by dividing the size of main memory available to users by the mean degree of multiprogramming. For paging systems, the size of memory allocated per job actually varies according to the current multiprogramming level of the SUC, but since jobs are assumed to be identical, the mean multiprogramming level of a paging system obtained over some fixed period of time is used. Furthermore for paging systems, it is assumed that the paging device can be either dedicated to paging activities, or shared between regular file I/O and paging activities. The memory reference characteristics of jobs and the page replacement policy of the operating system are assumed to interact with one another in a manner reflected by the programme lifetime function specific to the SUC [8]. It is also assumed that the operating system allocates memory on an equipartitional basis for the current number of jobs within the maximum multiprogramming level supported by the system.

OPTIMAL assumes each device i in the system has a finite number of alternative upgrade devices (referred to simply as alternatives). These are devices (usually faster or larger or both) which are candidates to replace device i or to be added to share the load of device i. In the case of replacement, a resale value for device i must be provided. In the case of addition, alternative j of device i is assumed to have the same set of upgrade alternatives as i. For each existing as well as alternative device, its capability, cost, and whether tuning is allowed are required as inputs. The capability of a non-memory device is assumed to be characterized by the service rate, and may be expressed in various terms. For example, for the CPU, it may be in number of millions of instructions per second (MIPS), or in number of jobs per second; for I/O devices, it may be in number of jobs per second or in number of page frames per unit time. The capacity, in terms of page frames or megabytes, is also needed for primary or secondary storage devices. Workload characteristics of SUC are expressed as visit ratios for I/O devices, mean number of instructions per job or visit ratio for CPU, and the multiprogramming level for memory.

2.2 THE ITERATIVE PROCEDURE

Some of the paramters used by OPTIMAL are listed below:

System Parameters:

Ncents : number of non-memory devices in the system

μ_i : service rate of device i

memsize : main memory size

[1] In the context of tuning, only on-line secondary storage I/O devices are modelled.

max-memsize : maximum main memory size supported by system

Workload Parameters:

z : mean thinktime for transactional workload

v_i : visit ratio, i.e., mean number of service requests made per job, for device i

μ_i : service rate of device i in requests per second

M : maximum multiprogramming level allowed in a memory-constrained system

m : mean size of memory allocated per job

$l(m) = lifetime(m)$: lifetime function for paging system

Economic Parameters

ΔM : unit increment on memory size

B : initial budget

$n\text{-}alt_i$: number of upgrade alternatives for device i

C_{ij} : cost of the jth upgrade alternative of device i

c_i : resale value of device i

B : initial budget

remain : running balance of the budget

OPTIMAL recommends tuning to be done initially if files are allowed to be moved across I/O devices. As mentioned earlier, balancing system load by re-assignment of files has been shown to improve system performance. An I/O device with very high utilization (close to 100%) may be a system bottleneck and improving the capability of other devices may not yield noticeable improvement in system performance. However, if files are re-assigned from this device to others balancing their service demands, utilization of the original bottleneck will be lowered. Thus, if tuning is allowed, it should be done before upgrading. Otherwise, upgrading the original system bottleneck alone is not likely to be the most cost-effective measure to improve system throughput.

After initial tuning (if possible), OPTIMAL operates as an iterative process.

Step 0 Balance load of I/O devices by reallocating files if possible. Find initial system throughput t.

The iterative process:

Step 1 For each device i for which upgrading is considered, and for each upgrade alternative j of device i, $0 < i \leq Ncents$, $0 < j \leq n\text{-}alt_i$:

Step 1.1 Try replacing device i by alternative j: if $C_{ij} > c_i$ and $remain \geq C_{ij} - c_i$, or if $C_{ij} \leq c_i$, substitute capability of alternative j for device i. Balance load of device i with other devices if possible. If replacement is not possible, go to Step 1.4.

Step 1.2 Evaluate system performance and find the throughput t^r_{ij} with respect to the replacement in Step 1.1.

Step 1.3 Restore capability of device i to the state in Step 1. If tuning was done in Step 1.1, restore visit ratios of tuned devices to the state in Step 1.

Step 1.4 Try adding alternative j to system: [2]

if $remain \geq C_{ij}$,
and if $i \neq$ memory and $Ncents < max$,
or if $i =$ memory and $memsize + \Delta M \leq max\text{-}memsize$,
then add a new device with capability equal to that of alternative j to the system. If i is an I/O device, move files to new device such that load is balanced across all I/O devices that can be tuned. If addition is not possible, go to Step 2.

Step 1.5 Evaluate system performance and find the throughput t^a_{ij} with respect to the addition in Step 1.4.

Step 1.6 Restore original system configuration by removing new device, and if $i \neq$ memory, restoring visit ratios to the state in Step 1.

Step 2 Calaculate the cost-effectiveness of each upgrading measure carried out in Step 1, i.e., for each device i, compute:

$$ceff^r_{ij} = (t^r_{ij} - t)/(C_{ij} - c_i)$$

and

$$ceff^a_{ij} = (t^a_{ij} - t)/C_{ij},$$

if the corresponding upgrade is possible.

Step 3 For device i and alternative j where

$$ceff^{action}_{ij} = \max_{i,j,action} \{ceff^{action}_{ij}\},\ action \in \{r, a\},$$

upgrade system by the corresponding *action*, i.e., either replacing device i by its j-th alternative, or adding alternative j to the system as a new device. Balance load of device i with other I/O devices if possible. Increment *Ncents* if *action* is addition of device to system.

[2] Addition of devices is only permitted for memory and for I/O devices for which files can be re-allocated.

Step 4 Record system throughput t with respect to upgrade decision in Step 3 (i.e. $t = t^r_{ij}$ or t^a_{ij}). Decrement *remain* by $(C_{ij} - c_i)$ if *action* is replacement, and by C_{ij} if *action* is addition.

The iterative process terminates when the remaining budget is insufficient for any type of upgrading action for any device. To take total disk capacity into consideration, when the iterative process terminates, the sum of the capacities of the disks in the recommended system is compared to the minimum required specified by the user. If the total in the system is equal to or exceeds the disk capacity constraint, the result is optimal and printed. Otherwise, OPTIMAL back-tracks to the most recent non-disk upgrade decision and replaces it by the most cost-effective disk upgrade. If the disk requirement is still not satisfied, this procedure is repeated until the disk capacity constraint is met or all of the non-disk upgrades have been revoked. In the latter case, there is no solution possible with the amount of money specified and a message is printed accordingly.

2.3 THE OUTPUT

OPTIMAL produces the following output:

1. A recommendation on how to distribute the given budget among the various upgrade alternatives, i.e., which devices to replace in the initial system and which devices should be added to the system.
2. The remaining budget.
3. The system throughput with respect to the recommended system configuration (after fine-tuning if possible).

3 IMPLEMENTATION ISSUES

3.1 SYSTEM PERFORMANCE ANALYSIS ALGORITHMS

Even though we have chosen to build system performance analysis tools for OPTIMAL based on Mean Value Analysis [13] [8], it should be pointed out that OPTIMAL is not dependent on these tools. In fact, given the appropriate analytic modelling tool, other systems, such as multiple class systems, can also be applied to OPTIMAL with minor modifications. Such modelling tools are widely available; for instance, BEST/1 [1] has been widely used in North America since 1979. QNETS [12] analyzes multiple class centralized systems with memory-constraint. Another package, QNAP2 [6], models and evaluates the performance of systems with varied complexities: open or closed systems with single or multiple job class, with or without memory constraint, with local or remote transactions over a network, and non-separable [3] systems with a finite number of processors linked together by a network.

Two notes on the exact Mean Value Analysis as applied to memory-constraint batch and paging systems. For memory-constraint batch systems where the total number of customers, *Ncusts*, is less than or equal to the multiprogramming level, M, of the system, rather than taking the mean throughput over all feasible populations 1,..., *Ncusts*, the system throughput is taken to be that obtained at multiprogramming level *Ncusts*. This is because all processes can be accommodated in main memory and there is no delay imposed outside of the multiprogramming subsystem (refer to Figure 1).

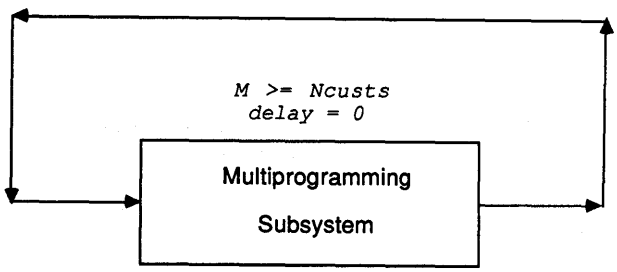


Figure 1: Closed Batch Model with Memory Constraint $M \geq Ncusts$

For paging systems, the visit ratio to the paging device for paging service [3] is dependent on the current multiprogramming level. Therefore, to calculate the system throughput at each feasible multiprogramming level, it is necessary to find the visit ratio and the service demand to the paging device for paging service at each of these levels. This visit ratio at any feasible population n can be derived from the formula:
$visit\ ratio_{pag} = service\ demand_{cpu} / lifetime(m)$,
where, according to the user-specified lifetime function, *lifetime(m)* yields the mean CPU time between consecutive page faults with respect to the memory allocation per job (m) when the current multiprogramming level is n (i.e., $m = memsize/n$).

3.2 OPTIMIZATION

Optimization is carried out in OPTIMAL in the following four main areas: saturation analysis and tuning, calculation of cost-effectiveness, optimizing multiprogramming level, and fine tuning.

[3] Paging devices may be shared between paging activities and regular file I/O activities.

3.2.1 SATURATION ANALYSIS AND TUNING. As mentioned in Section 2.2, a saturated device in a system is not necessarily the most cost-effective device to upgrade. If the saturated device is an I/O device where files can be moved to other I/O devices in the system to balance their service demands, then such tuning action may remove this device as the primary bottleneck. Some other device whose service demand has not been affected by the tuning measure may become the most heavily utilized device in the system and hence emerges as the current primary bottleneck. However, since our upgrade criterion is the ratio of the increased throughput to the cost of the upgrade, cost is also a factor in the decision process, and the most highly utilized device is not necessarily selected for upgrade.

3.2.2 CALCULATION OF COST-EFFECTIVENESS. If for some reason, the replacement cost of a new device is less than the resale value of an existing device in the system, OPTIMAL upgrades this device by replacing it with the alternative before proceeding to calculate the cost-effectiveness of the rest of the devices. The profit made in the upgrade transaction is accounted into the budget. In situations where the capability of an upgrade alternative is equal to or even lower than that of an existing device in the system, OPTIMAL ignores this upgrade possibility and proceeds to examine the other alternatives.

3.2.3 OPTIMIZING THE MULTIPROGRAMMING LEVEL. When memory is added to memory-constrained swapping systems, the multiprogramming level of the system may be increased up to the maximum possible, based on the mean memory size allocated per job for that system (see Section 2.1). System throughput will increase with the multiprogramming level for such systems. For memory-constrained paging systems, however, since throughput is convex with multiprogramming level [8], the optimal multiprogramming level is one at which system throughput is maximized. It is important to note that the optimal multiprogramming level is a function of the current workload and system configuration, including the lifetime function that characterizes the paging activities of the particular system.

As in the case of tuning, optimization of the multiprogramming level for paging systems should be done before the iterative process of OPTIMAL begins. This is achieved by evaluating the system at each feasible population from 1 to the current multiprogramming level, but halting as soon as the throughput obtained at level $M+1$ is less than that at M ($M+1 \leq$ current multiprogramming level). The initial optimal multiprogramming level is then set at M. At every step of 1.2 and 1.5 of the iterative process, optimization is also necessary. The optimization is carried out as outlined below.

If the current optimal multiprogramming level is M when an upgrade possibility is considered, evaluate the system at each multiprogramming level $M, M+1, \ldots, max\text{-}M$, obtaining system throughputs $t_M, t_{M+1}, \ldots, t_{max-M}$. The optimal multiprogramming level is the one corresponding to the maximum throughput rate. However, the evaluation process can stop as soon as $t_{M+n} < t_{M+n-1}$ for some $n > 0$ and $M+n \leq max\text{-}M$. In the latter case, the optimal multiprogramming level with respect to the specific upgrade possibility is $M+n-1$. Optimization is carried out in OPTIMAL so that at every step the most cost-effective upgrade measure is determined based on the *highest possible* throughput for each upgrade possibility.

3.2.4 FINE TUNING. Balancing service demands on I/O devices while keeping their service rates and the sum of their visit ratios constant may often result in non-integral visit ratios [4]. One goal of fine tuning is to obtain integral visit ratios based on the non-integral ones such that the throughput obtained with these visit ratios is at least as high as that obtained with the non-integral visit ratios.[4] As the relationship between throughput and visit ratios is intuitively complex, it is only reasonable to try integral visit ratios in the vicinity of the non-integral ones while keeping the sum of the visit ratios constant. Since the practicability of the *final* recommendations of OPTIMAL is the main concern, fine tuning is required only in the final step of OPTIMAL, rather than after every tuning measure.

4 PERFORMANCE OF OPTIMAL

To evaluate the performance of OPTIMAL, a control programme, CONTRL, was set up to enumerate all possible ways of allocating funds to the upgrading alternatives of a given system. Each possible way results in replacement of existing devices by upgrade alternatives and/or addition of new devices in the system. The optimal throughput produced by CONTRL is the maximum throughput over all such possibilities. The optimal throughput and the execution time are used as criteria for comparison between OPTIMAL and CONTRL. For systems with devices having two or more alternatives that are also candidates to be added to the system, and for a sufficiently large budget, it is obvious that the number of possible

[4]Empirical data from test examples on fine tuning have shown that throughput at least as high is achievable.

ways of allocating the budget is very large. Consequently, for most test systems, OPTIMAL performs far better than CONTRL with respect to execution time, and thus this criterion is of less interest here. The criterion of primary concern in such systems is the optimal throughput. A number of hypothetical systems of different complexities have been tested by both OPTIMAL and CONTRL. We present here four sets of typical results (Table 1) together with the input parameters for System 4 (Tables 2 and 3). In all the runs we have made, the improvement in throughput per dollar produced by OPTIMAL is always at least as good as that by CONTRL, and the absolute throughput is within 1.0% from the highest possible for each system.

	OPTIMAL	CONTRL	Conclusion
System 1			
User time	2.4	42.6	For timing,
System time	0.2	12.8	OPTIMAL is
Elapsed time	00:00:14	00:21:06	slightly superior.
Optimal throughput	0.303972	0.303972	Same
Amount spent	\$115000	\$115000	Same
Throughput increase per dollar spent ($\times 10^{-6}$)	1.343	1.343	Same
System 2			
User time	7.1	717.0	For timing,
System time	0.1	28.8	OPTIMAL
Elapsed time	00:00:26	00:57:12	is superior.
Optimal throughput	0.259581	0.261927	Differ by 0.89%
Amount spent	\$116500	\$126500	
Throughput increase per dollar spent ($\times 10^{-6}$)	0.632	0.601	OPTIMAL is superior.
System 3			
User time	57.7	3672.9	For timing,
System time	1.6	377.6	OPTIMAL is
Elapsed time	00:01:15	04:34:53	superior.
Optimal throughput	0.308968	0.311382	Differ by 0.775%
Amount spent	\$120500	\$125500	
Throughput increase per dollar spent ($\times 10^{-6}$)	0.928	0.910	OPTIMAL is superior.
System 4			
User time	83.6	26235.0	For timing,
System time	0.7	45.5	OPTIMAL is
Elapsed time	00:01:27	07:22:35	superior.
Optimal throughput	0.319416	0.321385	Differ by 0.616%
Amount spent	\$120500	\$130000	
Throughput increase per dollar spent ($\times 10^{-6}$)	1.014	0.955	OPTIMAL is superior.

Table 1: Comparison of Some Typical Results from OPTIMAL and CONTRL

- Closed interactive system with memory-constraint.
- Memory management: swapping.
- Disks 1, 2 and 3 can be tuned.
- *System configuration and workload:*
  - *Ncusts* = 35
  - z = 50.0s
  - B = \$130,000
  -

Device i	μ_i	v_i	c_i	disk capacity
CPU	250	621	\$ 5000	-
Disk 1	50	150	10000	142 (MB)
Disk 2	30	200	5000	71
Disk 3	40	120	4000	71
Drum	45	150	4500	-

- *Memory:*
  - *memsize* = 12MB
  - M = 10
  - *max-memsize* = 32MB
  - *max-M* = 12
  - c_{memory} = \$9000
  - minimum disk capacity required = 982MB

Table 2: System 4

Device i	Capability	Disk Capacity	Cost
CPU:	-	-	-
Disk 1:			
1	55	142	\$25000
2	70	280	40000
3	80	560	60000
4	60	280	30000
Disk 2:			
1	55	142	25000
2	70	280	40000
3	80	560	60000
4	60	280	30000
Disk 3:			
1	55	142	25000
2	70	280	40000
3	80	560	60000
4	60	280	30000
Drum:			
1	55	-	25000
2	70	-	40000
3	80	-	60000
4	60	-	30000
Memory:			
1	4	-	20000
2	6	-	30000

Table 3: System 4 Upgrade Alternatives

5 CONCLUSION

OPTIMAL, a software package which provides upgrade strategy based on maximum system throughput per dollar constrained by a given budget, minimum disk capacity requirement and available upgrade alternatives, has been implemented and described in the paper. Various optimization techniques, including optimizing the multiprogramming level and fine tuning, have been used to enhance its performance. Our results showed that OPTIMAL always gives throughputs within 0.0 - 1.0% of the absolute maximum for each system, and with equal or higher improvement in throughput per dollar spent as compared to that obtained with the absolute maximum throughput. OPTIMAL is interactive, prompts the user for input data, and is very easy to use.

OPTIMAL can be easily enhanced to incorporate a more sophisticated set of system performance analysis algorithms, thus increasing its scope of application to cover more complex systems, such as multiple job class systems, and systems with elaborate I/O subsystems.

5.1 MULTIPLE JOB CLASS SYSTEMS

For systems in which the jobs being modelled have significantly different behaviours (for instance systems with a mixture of CPU and I/O bound jobs), a multiple class model can provide more accurate results. To include multiple job class systems, it is only necessary to modify OPTIMAL to accept the workload characteristics of each job class and to use exact MVA that models multiple job class systems in steps 0, 1.2, 1.5 and 4 to obtain throughput values for each job class which are then summed to produce the system throughput.

5.2 I/O SUBSYSTEMS

Because of advances in VLSI technology, processor, main memory and other purely electronic components of the computer system have outpaced the advancement in I/O subsystems that use rotating device technology. Consequently it is not uncommon for current systems to have bottlenecks in I/O components rather than CPU or main memory. A common I/O subsystem consists of various elements of varied intelligence: channels, controllers, string heads, disks. Being able to locate the bottleneck element in a highly utilized I/O subsystem is a key to improving system performance. The current implementation of OPTIMAL captures the details of an I/O subsystem in the disk service demand obtained from measurement data. Thus a more sophisticated modelling approach, such as the one suggested in [8], is necessary to account for the effect of improving each I/O subsystem component on system performance.

6 APPENDIX

>optimal [5]

```
**************************************
*    System and Workload Parameters    *
**************************************

Enter number of customers: 40
Enter average think time of customers (sec.): 40.0
Enter number of devices excluding memory: 3
Is there a paging device? (0 = 'no', 1 = 'yes') 0
Is memory to be evaluated? (0 = 'no', 1 = 'yes') 1

For the following, device 1 must be the CPU
and device 4 main memory

Enter number of disks used for file services only: 2
Enter their device numbers: 2, 3

Is there a minimum disk capacity to be
met? (0 = 'no', 1 = 'yes')1
Enter minimum disk capacity required (e.g.MB): 982

                *********
                *  CPU  *
                *********

Enter visit ratio for device 1 : 271

Enter service rate for device 1 (transactions/sec.)
:100

          ***********************
          *  On-line I/O Device  *
          ***********************

Enter visit ratio for device 2 : 120
Enter service rate for device 2 (transactions/sec.)
:50
Enter disk capacity for device 2 (e.g. MB): 142
```

[5] Invoke OPTIMAL.

```
          ************************
          *  On-line I/O Device  *
          ************************

Enter visit ratio for device 3 : 150
Enter service rate for device 3 (transactions/sec.)
:40
Enter disk capacity for device 3 (e.g. MB): 71

                ************
                *  Memory  *
                ************

Enter maximum memory size possible
(e.g.KB, page frames): 32
Enter maximum degree of multiprogramming supported:
15
Enter main memory size (e.g.KB, page frames): 16
Enter mean degree of multiprogramming: 8

                ************
                *  Tuning  *
                ************

Enter number of devices on which tuning is allowed:
2
Enter their device numbers: 2, 3

*****************************
*    Economic Parameters    *
*****************************

Enter budget (dollars): 130000

Enter market value of device 1 (dollars): 5000
Enter no. of alternatives available for upgrading
device 1: 0

Enter market value of device 2 (dollars): 10000
Enter no. of alternatives available for upgrading
device 2: 3
For alternative 1 for device 2,
enter its capability (transactions/sec.): 50
enter its price (dollars): 25000
enter its disk capacity (e.g. MB): 142
For alternative 2 for device 2,
enter its capability (transactions/sec.): 60
enter its price (dollars): 30000
enter its disk capacity (e.g. MB): 280
For alternative 3 for device 2,
enter its capability (transactions/sec.): 80
enter its price (dollars): 60000
enter its disk capacity (e.g. MB): 560

Enter market value of device 3 (dollars): 5000
Enter no. of alternatives available for upgrading
device 3: 3
For alternative 1 for device 3,
enter its capability (transactions/sec.): 50
enter its price (dollars): 25000
enter its disk capacity (e.g. MB): 142
For alternative 2 for device 3,
enter its capability (transactions/sec.): 60
enter its price (dollars): 30000
enter its disk capacity (e.g. MB): 280
For alternative 3 for device 3,
enter its capability (transactions/sec.): 70
enter its price (dollars): 45000
enter its disk capacity (e.g. MB): 280

Enter market value of device 4 (dollars): 3000
Enter no. of alternatives available for upgrading
device 4: 2
For alternative 1 for device 4,
enter its capability (e.g.KB, page frames): 2
enter its price (dollars): 15000
For alternative 2 for device 4,
enter its capability (e.g.KB, page frames): 4
enter its price (dollars): 20000

********************
*   Adjustments    *
********************

Adjustments and corresponding adjustment codes:
*******************************************************************************
*  1 * number of customers          * 11 * max. degree of                     *
*    *                              *    * multiprogramming                   *
*  2 * mean thinktime               * 12 * current main mem. size             *
*  3 * number of devices            * 13 * mean degree of                     *
*    *                              *    * multiprogramming                   *
*  4 * system is/ not paging        * 14 * budget                             *
*  5 * system is/ not memory-       * 15 * market value of device             *
*    * constrained                  *    *                                    *
*  6 * instruction execution        * 16 * number of upgrade                  *
*    * rate of CPU / service        *    * alternatives for a device          *
*    * rate of a device             *    *                                    *
*  7 * visit ratio of device        * 17 * lifetime function                  *
```

```
*  8 * paging device is/ not shared * 18 * file service disk devices     *
*  9 * no.of tuneable devices       * 19 * disk capacities              *
* 10 * max. memory size             * 20 * no further adjustments       *
*************************************************************************
Enter adjustment code: 20

***Processing begins ... optimal solution is ...

                      **********************
                      *  Recommendations   *
                      **********************

*************************************************************************
* Optimal system throughput (not fine-tuned):   0.368574               *
* Optimal system throughput (fine-tuned):       0.368574               *
*-----------------------------------------------------------------------*
* Recommended system configuration:                                     *
*-----------------------------------------------------------------------*
* Number of added non-memory devices:              3                    *
* Their device numbers:                            4                    *
*                                                  5                    *
*                                                  6                    *
*                                                                       *
*************************************************************************
* Device *  EqD+  *   Service Rate  *   Visit Ratio   *  Amount Spent   *
*************************************************************************
*   1    *   1   *        100.00    *       271       *        0.00     *
*   2    *   2   *         50.00    *        50       *        0.00     *
*   3    *   3   *         40.00    *        40       *        0.00     *
*   4    *   2   *         60.00    *        60       *    30000.00     *
*   5    *   2   *         60.00    *        60       *    30000.00     *
*   6    *   2   *         60.00    *        60       *    30000.00     *
*************************************************************************
* Recommended memory size:                  24       *    40000.00     *
* Total disk capacity achieved:             1053     *                 *
*************************************************************************
* Total budget required:                                  130000.00     *
* Remaining budget:                                         0.00        *
*************************************************************************

+ EqD is the existing device for which *Device* is an alternative.

Upgrade completed.
```


Acknowledgement

The work reported in this paper was partially supported by the Canadian Natural Sciences and Engineering Research Council under Grant A3554.

References

[1] *BEST/1 User's Guide*, Release 5.0, BGS Systems, Inc., Lincoln, MA, Aug. 1979

[2] J.C. Cooper, *A Capacity Planning Methodology*, IBM Systems Journal, Vol.19, #1, 1980, pp.28-45

[3] P.J. Denning, J.P. Buzen *The Operational Analysis of Queueing Network Models*, Computing Surveys, Vol.10, #3, Sept. 1978, pp.225-261

[4] D. Ferrari, G. Serazzi, A. Zeigner, *Measurement and Tuning of Computer Systems*, Prentice Hall, 1983

[5] L.F. Hodges, *Workload Characterization and Performance Evaluation in a Research Environment*, Proc. of the 1982 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Vol.11, #4, 1982

[6] *New Users' Introduction to QNAP2*, Version 08/84, INRIA, France, 1984

[7] S.R. Kumar, R.B. Lake, C.T. Nute, *File Allocation Methodology for Performance Enhancement*, Proc. 16th CPEUG Meeting, 1980, pp.175-188

[8] E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik, *Quantitative System Performance, Computer System Analysis Using Queueing Network Models*, Prentice Hall, 1984

[9] A.P. Levine, *ESP: an Expert System for Computer Performance Management*, CMG '85 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, 1985, pp.181-186

[10] T.L. Lo, *Computer Capacity Planning Using Queuing Network Models*, Proc. of Performance '80 the 7th IFIP W.G. 7.3, International Symposium on Computer Performance Modelling, Measurement and Evaluation, Toronto, Ont., May 1980

[11] A.T. Maeda, L.A. Aho, *Using Expert System Technology to Evaluate System Resource Planning Alternatives: a Management Perspective*, CMG '85 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, 1985, pp.558-563

[12] J.F. Pang, *Characterizing User Workload for Capacity Planning*, M.Sc. Thesis, U. of British Columbia, 1986

[13] M. Reiser, S.S. Lavenberg, *Mean Value Analysis of Closed Multichain Queueing Networks*, JACM, Vol.27, #2, April 1980, pp.312-322

[14] R.A. Orchard, *Algebraic Models for CPU Sizing*, CPEUG 83 Proc. of the Computer Performance Evaluation Users Group 19th Meeting, San Francisco, CA, Oct. 1983, pp.116-134

[15] J.M. Ostrowski, J.L. Webb, T.A. Hilton, *Implementing Expert System Technology into the Field of Computer Performance Evaluation*, CMG '85 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, 1985, pp.245-250

[16] K.O. Salawu, *Capacity Prediction for UNIX Systems*, CMG '85 Proc. of the International Conference on the Management and Performance Evaluation of Computer Systems, 1985

[17] K.C. Sevcik, G.S. Graham, J. Zahorjan, *Configuration and Capacity Planning in a Distributed Processing System*, Proc. 16th CPEUG Meeting, 1980, pp.165-171

ESTABLISHING USEFUL BUSINESS RELATIONSHIPS

Dr. Leilani E. Allen
Amdahl Corporation

ABSTRACT

Capacity Managers (CM) are constantly placed in the position of negotiation among conflicting interests—be it with vendors, users, peers or supervisors. Thus, it is essential that CM professionals acquire this essential management skill. The key to be a better negotiator lies in understanding basic organizational principles and using them to analyze behavior. This paper describes these principles and shows how they may be used by the CM in a negotiation process.

INTRODUCTION

One of the most difficult problems that CM's face is the need to establish effective business relationships with user departments. More so than many other areas within Information Systems (I/S), CM's must actively seek information about the business to include in their forecasts and projections. They must keep users apprised of the data center's plans, and must often act as the data center's representative in project meetings. CM's very often serve as leaders within I/S as well, getting various areas to agree on longer-term strategies and directions.

To be successful, CM's must acquire solid management skills, including the ability to understand how organizations operate. They must understand what motivates their colleagues to act the way they do. They must learn to work within the organizational culture to achieve their aims. In short, CM's must understand and become adept at organizational politics.

This is especially critical if we understand that probably the most fundamental task of management is negotiation. Negotiation is here defined as "the process for achieving consensus to a common goal, plan or strategy." This is really the key to building more effective business relationships. CM's in particular must learn how to negotiate with peers, superiors, users and external vendors. And negotiation is a learned skill.

This paper is written for managers and planners who seek to improve their negotiation skills. It begins with a discussion of two basic philosophical models that serve to guide human behavior. It then discusses one of these models in greater detail, describing two fundamental organizational principles that can be applied in the context of the corporate culture. Then it describes the negotiation process in detail, illustrating the points by means of a hypothetical service level agreement (SLA) negotiation.

ORGANIZATIONAL MODELS

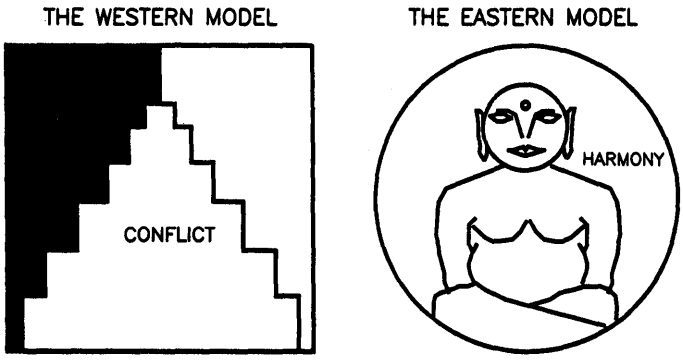
One of the distinguishing hallmarks of any culture is its belief system concerning the way the universe functions. This is as true for the United Nations as it is for the Bushmen of the Kalahari or Acme Enterprises. From this belief system comes a set of norms, or behaviors, that are tolerated within the culture.

The belief system usually includes a philosophical mode. Most models are fairly simplistic and can be summarized in a few key phrases. This is precisely what gives these models their power—they do not require extensive analysis or intellectual effort.

Two fundamental models have been with us since the dawn of civilization. One arose in the East, in Asia, and was based on the view that the universe was a whole.

Man was simply a part of the whole (and sometimes not even the most important part.) the goal in this belief system was to seek harmony between the various elements of the universe--Harmony with Nature, Harmony with God, Harmony within Oneself, Harmony with Others. The fundamental behavior was to seek "cooperation" to maintain this harmony.

FIGURE 1: TWO PHILOSOPHICAL MODELS



The other model arose in the West, in Mesopotamia. The Babylonians and Sumerians viewed the universe as divided into the opposing forces of Light and Darkness. Each side struggled for supremacy, with human beings becoming involved in the struggle almost as pawns. This Conflict Model was adopted by most major Western belief systems, variously characterized as God versus the Devil, Man versus Nature, Democracy versus Autocracy, the Good Guys versus the Bad Guys, Us against Them. It is still the fundamental organizing principle for most Western institutions.

I wish to emphasize that either model can be applied in any culture. Indeed, it can be argued that Sweden has largely adopted the Cooperative model, while the Chinese since 1948, have largely operated on the basis of the Conflict model. Clearly, one is not more valid than the other. In this country, many of the analysts of Japan, Inc., or the gurus of excellence point to the benefits to be derived from viewing the marketplace (and certainly the enterprise) as a vast, interconnected whole in which

conflict avoidance and mutual advantage are the guiding principles for behavior.

However, most of us still work in business settings in which the Conflict model is the norm. Therefore, analyzing its key beliefs in more detail is an essential starting point for our discussion of business relationships.

THE CONFLICT MODEL

Two of the most fundamental principles of organizational behavior arising from the Conflict model are:

1. The Balance of Power (BOP).
2. The Iron Law of Oligarchy (ILAW).

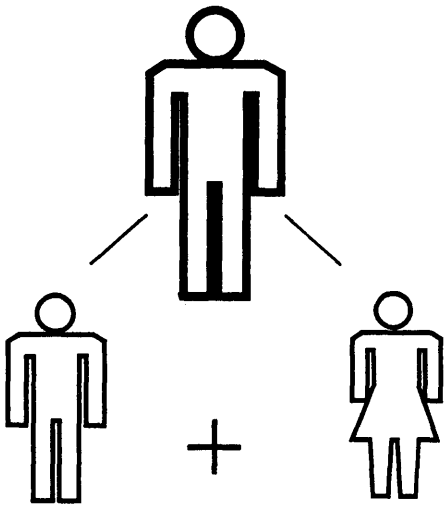
THE BALANCE OF POWER

The Balance of Power (BOP) concept states that we all live in a world of scarcity--that there are never enough resources, goods, wealth or prestige available to satisfy everyone. Therefore, conflicts will arise over who should control these scarce commodities. The degree of control an individual, group or national entity exercises over these commodities is a direct reflection of that entity's power.

In the BOP concept, such issues as friendship or morality should have nothing to do with the contest between entities. Rather, each entity is motivated to act on the basis of its perceived interests. It is not wrong or immoral to seek to strengthen one's position of power; it is the natural way of things. We should not be surprised by tactics of aggrandizement; rather, we should expect them and be able to counter them to our own advantage. An "enemy" is simply an entity who is seeking to acquire commodities to our disadvantage. A "friend" is another entity who is as interested as we are in stopping (or at least minimizing) that effort.

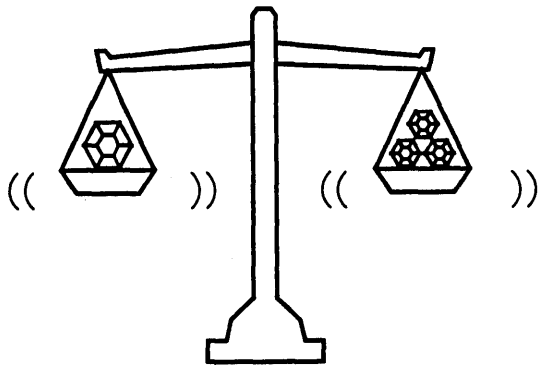
This is the basis of strategic alliances. Two entities who might have nothing much in common can join together to counter the moves of a third party. This alliance is made purely on the basis of mutual interests. We do not have to like or approve of our alliance partner. Indeed, we both recognize that, given the shifting nature of events, we will probably find ourselves on opposite sides in the next controversy.

FIGURE 2: STRATEGIC ALLIANCE



This is where the concept of "balance" comes in. BOP sees the world as a multi-armed scale, with one set of entities and the commodities they control balanced against the other sets. When one entity upsets the balance, all rational participants seek to reconfigure themselves in such a way as to restore the balance. This can mean seizing or neutralizing the rogue entity's resources, undermining its strategic alliances or presenting a show of force.

FIGURE 3: MAINTAINING THE BALANCE



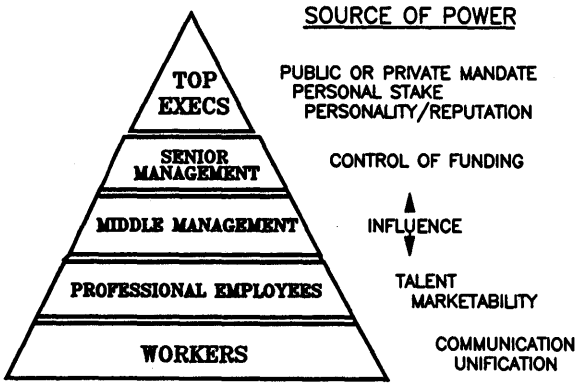
Lest this sound like a totally bloodthirsty approach, consider that BOP always prefers negotiation to war. Wars have a terrible effect on the balance of things and they destroy scarce commodities. Victories can be Pyrrhic; the long-term consequences of war are hard to predict. Negotiations are much more controlled and almost always lead to outcomes that restore the BOP and order.

Consider too, that although BOP denies that morality should be the basis of interaction (because morality is subjective), it also denies that prejudice should play a role. Thus, anyone who can influence the BOP should be a part of the negotiation, however humble they might be. BOP also seeks to preserve all the players in the game, because power vacuums are dangerous. The successful player recognizes these basic rules and uses them to full advantage in negotiation.

IRON LAW OF OLIGARCHY

Iron Law of Oligarchy (ILAW) finds that, in any organization, a power structure exists. At the tip of the power structure is an elite, and it is the elite which actually rules. This is as true for democracies as the most rigid totalitarian state; as true for "matrix organizations" as the conventional corporate hierarchy. Any mass organization will, over time, come to be ruled by a small elite or oligarchy.

FIGURE 4: THE POWER STRUCTURE

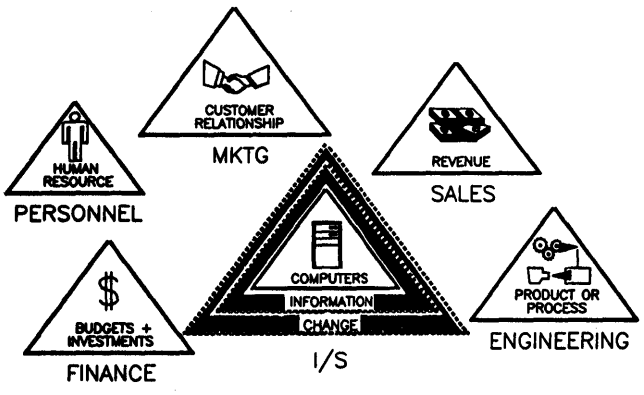


Now, organizations differ considerably on how they select these elite few, and what kind of constrictions are placed on them once they are in power. In more democratic organizations, the masses are formally elect representatives which have been carefully preselected by the elite from among their own ranks. In other organizations, different groups within the elite battle for positions of preeminence. Some rulers are explicitly recognized as such, others are "powers behind the throne." Yet, in no organization are the majority of decisions made by the masses.

This principle requires us to conclude that the average person has little to do with decision making. It can lead to cynicism and inaction. However, the positive side to this law is the fact that for any given decision, the number of decision makers involved is usually quite small. Certainly the numbers are small enough that we can identify the decision makers and seek to influence them. Thus, if we look at change as an incremental process, involving a slow series of small but significant decisions, it should be clear that we can position ourselves to be part of that decision-making process.

There are any number of ways of doing this. Some cultures admit members to the elite only by way of birth or sometimes marriage. Other cultures include or exclude members based on race, sex, religion, education, age or wealth. Most corporations are meritocracies, in which membership in the elite is a function of particular skills or talents, usually defined as the ability to make money for the enterprise. Most government agencies use longevity as the key criterion for entry into the elite.

FIGURE 5: POWER BASES



Corporate cultures differ as to the stress they put on certain skills or talents over time. Originally, the ability to engineer or manufacture products was the key skill. Nowadays, it is the ability to manage corporate finances which seems to be the key ingredient to the top. In the future, it may well be the ability to manage change.

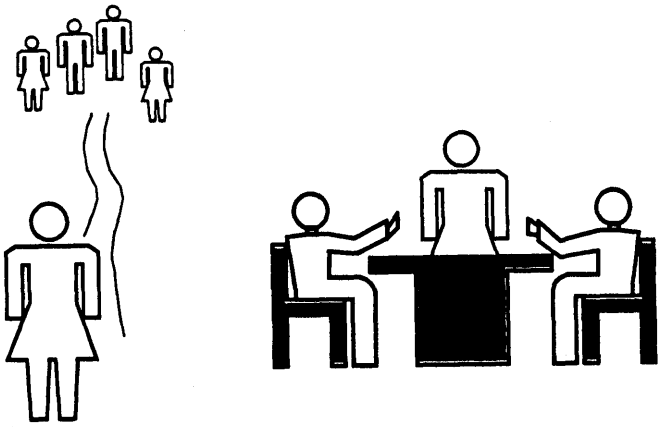
CAPACITY MANAGEMENT

Capacity management professionals can use these organizational principles in a number of constructive ways. First, they should be able to identify I/S's base of power and how that power base affects its role in the organization. They can realize that the source of conflict between their own department and other areas of the company has much less to do with technology than with the desire to control scarce and valuable resources. Technology is never neutral--it changes the way people are organized, the way they work, the way they make decisions. Other departments naturally see this as a threat to their own position within the power structure.

On the positive side, CM's should use BOP to analyze the operational departments more carefully, identifying opponents and potential allies. They should see that given the issue at hand, the roster of allies versus enemies can shift. Therefore, a fair amount of constraint and demonstrations of goodwill are called for, so as not to irrevocably antagonize any entity. Most good or universal truth will do little to marshal support for their plans, but that appealing to mutual interests is far likelier to ensure success.

CM's have two ways of responding to ILAW. They can seek to become part of the decision-making elite by demonstrating their business acumen, or they can maximize their specialist talent and become a trusted advisor to that elite. Both paths have advantages and disadvantages. But the choice always begins with understanding the organization's culture, identifying the oligarchy, and acting in such a way as to support the oligarchy's visions and goals.

FIGURE 6: MEMBER OR ADVISOR?



Obviously, I/S is not alone in seeking to enter or influence the oligarchy. Most savvy employees are attempting to gain the elite's recognition and approval. This usually takes the form of successfully developing and implementing programs and projects. This requires resources. Thus, there is always a contest for power in organizations, usually involving the acquisition and control of resources. Resolving these power conflicts is the function of managers through a process of negotiation.

NEGOTIATION

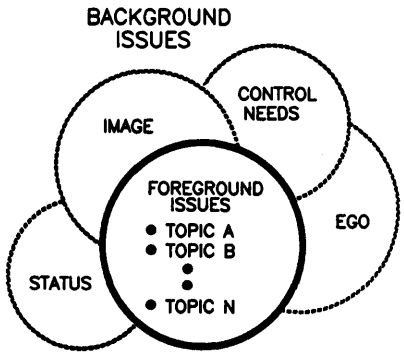
BASIC PRINCIPLES

We are constantly involved in negotiations of one kind of another, whether in our personal or professional lives. Yet, it is curious how seldom we take a rigorous approach to the subject. Somehow, being a good negotiator is viewed in a negative light, as if only slippery characters or Machiavellian mentalities were truly good at it. But negotiation, like so much else, is a learned skill. Moreover, it is the essential skill of management.

How can we improve our odds in any negotiation? There are two key principles to apply. The first is to understand that any negotiation is a joint enterprise, not a solo effect. The very heart of negotiation is a give and take between two or more parties. Thus, preparation for the negotiation must focus on not only one's own requirements and aspirations, but those of the other party as well.

In this process, it is important to distinguish between tangible and intangible outcomes. That is, a negotiation may have specific points or topics that need to be resolved. These are the foreground issues--"What should response time be?" "What specific performance issues will be included in the system requirements document?" "What factors will be included in the CPU purchase agreement?"

FIGURE 7: TANGIBLE AND INTANGIBLE OUTCOMES



The intangible outcomes revolve around the ego or psychological needs of the participants. These background issues include, "How will I look to my peers or my boss??" "How important is it to appear to be in control of the situation?" "How necessary is it to have decisions appear as if they came from the other side?" A truly successful negotiator spends as much time worrying about perception as about reality.

Inattention to this principle has resulted in many people coming away from a negotiation with a sense of victory only to find that their presumed adversary is actively taking steps to negate the outcome. In this case, the win has been strictly on the tangible side. The opponent's commitment to the solution has not been achieved. The performance analyst does not want to negotiate such tough requirements that the applications designer will use this as an excuse for project delays. The planner should not impose such penalties on users in a Service Level Agreement (SLA) negotiation that they feel abused and free to denigrate the department.

The second basic principle to remember is that real life rarely affords a demonstration of a WIN/LOOSE scenario. Almost all outcomes are compromises. Again, compromise may be viewed in a negative light. but the alternative to compromise is statis, a condition of complete immobility. WIN/LOSE games are tantalizing but they are rarely the means to long-term success.

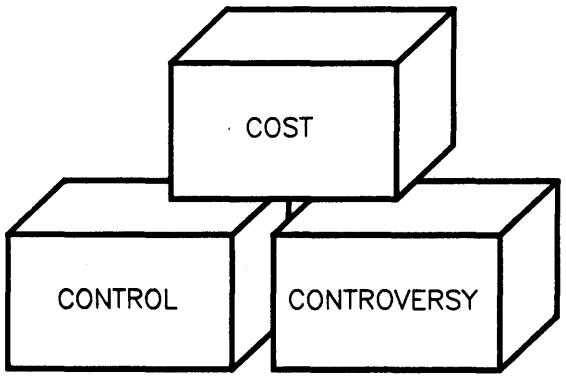
A METHODOLOGY

There are a number of specific steps that can be taken to encourage more successful outcomes using these two principles. Step One is to establish the true topic of the negotiation. That is often not as easy as it sounds. What appears to be a simple agreement on IMS response time and availability might actually be the first step toward a chargeback system. The negotiation may involve obtaining support and concurrence on the new policy. The user representative may be apprehensive about the long-term consequences of this change, and may feel that concessions made now will set unwelcome precedents.

The best negotiations are straight forward and focused. It may be necessary to explicitly state that each negotiation is a separate deal and that any agreement made now will not be used in future agreements. This should increase the flexibility on both sides.

Step Two is to identify all the necessary parties to the negotiation, that is, anyone who has a vested interest in the outcome. Not including participants means running the risk of having the outcome overruled. For our IMS SLA, users or project managers may seek to sell their viewpoint to higher ups and undermine the CM. Rather than gnashing teeth over the underhandedness of this approach, one should adopt it. An understanding of both the formal and informal corporate structure should include knowing who will ultimately commit to the agreement and what is likely to influence the decision.

FIGURE 8: THE THREE Cs'



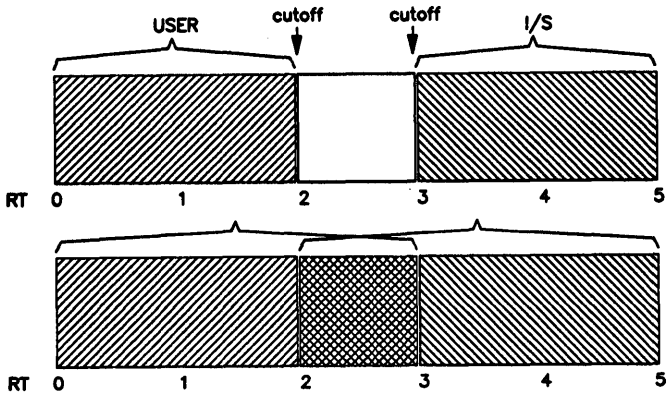
Typically, decision makers are concerned about the three "C's", Cost, Control and Controversy. A given project or program should meet certain cost/benefit criteria as established by the enterprise. For instance, the SLA should provide tangible benefits in the form of better workload forecasts to assist in planning upgrades, or more reliable response times to allow the users to manage their workflow. The project proposal should include sufficient mechanisms and controls to ensure that management has a reasonable chance of achieving a desirable outcome. Essential controls in an SLA include the ability to identify and explain noncompliance with the service level objectives.

Finally, the project should not create undue controversy--it should not seriously threaten organizational efficiency, effectiveness or public image. I/S management must be convinced that the SLA will smooth or regulate relations with the users and will allow the department to improve its image. User management must believe that the SLA opens a communications channel to I/S to allow their particular needs to be satisfied.

Having identified the participants, it is necessary to establish and rank their tangible and intangible objectives. Clearly, the greater the knowledge of the other side, the easier this will be. Sometimes some informal fact finding with colleagues or associates in other companies is appropriate. It is extremely difficult to negotiate with strangers. For the SLA, some research into the personalities involved, their "hot buttons", particular strengths and weaknesses and special situations would be recommended.

Once the issue lists are completed, the Fourth Step is to draw a line at each side's cutoff point. This is the place beyond which one could not compromise, the true bottom line for any negotiation. If the bottom lines are mutually exclusive, serious negotiation at this stage is probably a waste of time. It will then be necessary to go back to management to authorize changes in terms.

FIGURE 9: CUTOFF POINTS

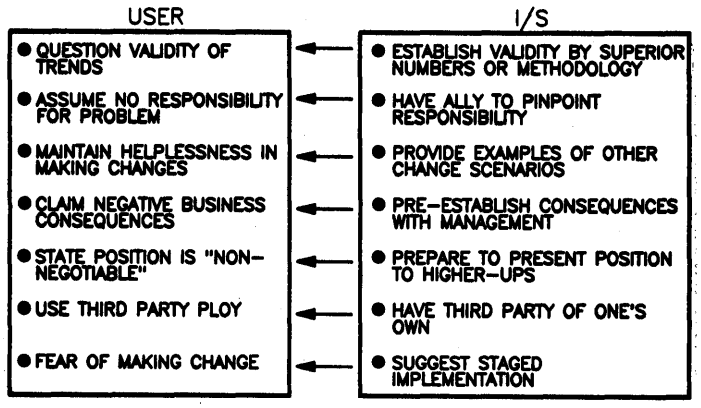


For instance, the user department might have made commitments to external customers which require two seconds or less response time for 90% of all transactions during peak periods. The CM knows this goal cannot be achieved without upgrading the CPU, an option which his/her management has absolutely rejected during this fiscal year. The negotiating positions are mutually exclusive. I/S and user management must create some flexibility in their own positions for the negotiation to work.

If there is still room for maneuvering, an assessment of strength should be prepared. This denotes what forces each side can bring to bear to win the issue. This might include such things as budget, personnel, time, management commitment, support from third parties, better information or more experience. In an SLA, I/S brings knowledge of the capabilities of the hardware and software management commitment and trained personnel. The user might bring support from third parties and budget clout.

The final preparatory step is to prepare a list of tactics. For instance, "How do I intend to establish certain facts or support my demands?", "How can I defuse the negative aspects of this position?", "How can I effectively play off one side against the other?" An equivalent list of tactics should be made for the other side, along with notations on how each of his tactics will be countered.

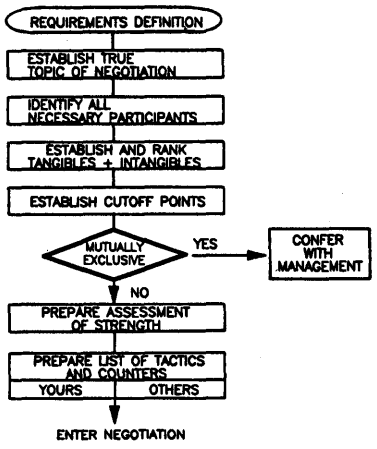
FIGURE 10: TACTICS AND COUNTERS


USER
I/S
● QUESTION VALIDITY OF TRENDS
● ESTABLISH VALIDITY BY SUPERIOR NUMBERS OR METHODOLOGY
● ASSUME NO RESPONSIBILITY FOR PROBLEM
● HAVE ALLY TO PINPOINT RESPONSIBILITY
● MAINTAIN HELPLESSNESS IN MAKING CHANGES
● PROVIDE EXAMPLES OF OTHER CHANGE SCENARIOS
● CLAIM NEGATIVE BUSINESS CONSEQUENCES
● PRE-ESTABLISH CONSEQUENCES WITH MANAGEMENT
● STATE POSITION IS "NON-NEGOTIABLE"
● PREPARE TO PRESENT POSITION TO HIGHER-UPS
● USE THIRD PARTY PLOY
● HAVE THIRD PARTY OF ONE'S OWN
● FEAR OF MAKING CHANGE
● SUGGEST STAGED IMPLEMENTATION

For example, the user might charge that response time has traditionally been bad and is getting worse, causing projects to slip. The CM should know the performance history of the application and be able to challenge the validity of this claim and point to other possible causes. The users might maintain that they have no control over when work is submitted to the computer. I/S should investigate the work processes in that department to determine whether control is impossible or just untried. Here the CM might offer to negotiate with a third party for assistance.

In this way, it is possible to rehearse the process in advance and the surprise factor is minimized. This will allow greater control of the situation, which means better odds of achieving goals. Even if all the goals are not achieved, the other side is likely to be impressed by the CM's thoroughness and coolness under pressure.

FIGURE 11: NEGOTIATION STRATEGY



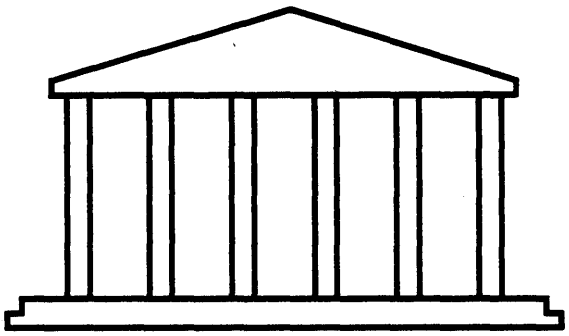
BUSINESS RELATIONSHIPS

Clearly, negotiating is a learned skill. These preparations can be complicated and time consuming the first time through. But with time they can become almost unconscious. The key to success is

understanding that opposition to one's plans and ideas is not the result of malice or stupidity. It is a natural consequence of a world view that sees conflict as the basic mode of human interaction.

The message of this paper is that this model, and the organizational principles that flow from it can be used in a positive way to build effective business relationships with our colleagues. The key is successful negotiation where conflicts are resolved in a mutually satisfactory way. Justice Louis Brandeis has observed,

"Harmony in...life is a resultant of the struggle between contending forces. In frank expression of conflicting opinion lies the greatest promise of wisdom."



"Harmony in...life is a resultant of the struggle between contending forces. In frank expression of conflicting opinion lies the greatest promise of wisdom."

– Justice Louis Brandeis

A SYSTEMS ENGINEERING LOOK AT SAA

James R. Cook, Managing Partner

Data Strategies, Systems Management Consultants

ABSTRACT

Most experienced I/S planners pay considerable attention to vendors' statements of direction. Architecture statements, which are much rarer, are even more valuable as planning guides, because they represent a technical plan as opposed to a marketing plan. If history is any guide, many of us will find IBM's Systems Applications Architecture occupying much of our professional lives for the next several decades, just as our counterparts of fifteen years ago became obsessed (or possessed) by SNA.

Recognizing that many of the implications of SNA could have been predicted from the original descriptions of the architecture, I have applied the techniques I generally use to assess competing products for vendors to the limited information currently available on SAA. Based on this analysis, I have developed some startling (and surely controversial) predictions, among them:

- The end of the System/370 architecture,
- The obsolescense of 3270-type terminals, and
- The "demise" of CICS!

I hope the readers find the conclusions as thought-provoking as I do.

1. INTRODUCTION

"...the SAA announcement was by far...IBM's most important statement in a decade....More than any new product, SAA defines the future for IBM and its customers."

David Kull, *Computer and Communications Decisions*

Most experienced I/S planners pay considerable attention to vendors' statements of direction. Of course, it makes good sense to pay attention when vendors tell you what they plan to do in the future. Architecture statements, which are much rarer, are even more valuable as planning guides because they represent a technical plan as opposed to a marketing plan.

In recent history, IBM has only made two major architecture statements: SNA in 1974 and this year's System Application Architecture announcement. Various industry analysts have claimed that SAA is everything from marketing hype to an attractive but unlikely pipe dream.

I see SAA as more than any of these things. Beyond the narrow limits of the announcement, I see a reflection of irrefutable technological and economic forces which will significantly change mainframe computing as we know it. I believe that the support for my beliefs has been made numerous time, in print, by various IBM developers. I have attempted to bring these statements together within the framework of the SAA announcement and to that end have extensively quoted a number of articles and authors.

I expect my comments to be controversial; after all, they imply the obsolescence of many vendor products (both IBM's and others). Since no ones crystal ball is perfectly clear, I recommend that you read my analyses and draw your own conclusions. But I remind those who discount SAA as a major force that, when originally announced, IBM's System Network Architecture was only supported by the DOS/VS operating system and the only SNA terminals were on the 3600 Finance Communication System.

2. WHAT IS SAA?

"Systems Application Architecture is a collection of selected software interfaces, conventions, and protocols....[It] will be the framework for development of consistent applications across...future offerings..."

Systems Applications Architecture An Overview

Opinions differ as to just what SAA is. There is the established viewpoint of industry analysts, and there is a considerably broader view (held by myself) which is outlined in the following section. Of course, neither of these viewpoints is important. The important view is held by IBM and that will not be revealed until it makes good marketing sense.

2.1. THE COMPONENTS OF SAA

Like most architectures (e.g. SNA), SAA is most easily described as a series of related layers which will be implemented across each of the applicable operating environments. These layers are:

- The Common Programming Interface
- The Common User Interface
- The Common Communications Support
- The Common Applications Library

The contents of and relationships between these layers are shown in Figure 1.

2.1.1. SAA's OPERATING ENVIRONMENTS. SAA is specifically directed toward "...the three major IBM computing environments..."[1] which are defined as:

- o System/370
  - TSO/E under MVS/XA
  - CMS under VM
- o System 3X
- o Personal Computer
  - Operating System/2

I prefer to think of these environments as: mainframe systems, departmental (small-work-group) systems, and individual systems because the functional requirements at each level are more indicative of the nature of operation than are the operating systems. For example, a 9370 has more in common functionally with a System 38 than with a 3090 in spite of the fact that they may both run MVS.

2.1.2. THE COMMON PROGRAMMING INTERFACE The components of The Common Programming Interface fall into two categories: common languages and common services. SAA will contain specifications for each component and applications which conform to these specifications will operate consistently under any of the three environments.

The currently supported languages are: C, COBOL, and FORTRAN supplemented by an Application Generator based on Cross System Product and a Procedures Language based on REXX.

The Services consist of Structured Query Language (SQL) for database access, a dialog interface based on the current implementation of the EZ-VU product for the PC, presentation services which are based on S/370 GDDM but clearly borrow heavily from Microsoft Windows, and a report generator based on S/370 Query Management Facility (QMF).

2.1.3. THE COMMON USER INTERFACE The user interface specification will provide commonality in three categories physical, syntactical, and semantic. Physical consistency will ensure that the user interface hardware (e.g. keyboards, displays, etc.) will have the same physical layout. In other words, you can expect your PC to have a keyboard suited to a mainframe for the foreseeable future.

Syntatical consistency will ensure that the location and sequence of presentation of information on displays will be the same across systems and applications. It will also "force" keystroke sequences controlling generic functions to be identical. The current implementation of PF5/6 for Page-Up and Page-Down when using the PS/2 configurator, as opposed to PF7/8 as implemented by ISPF, or PgUp/PgDn as labeled on the keyboard, is an example of lack of syntactical consistency.

Semantic consistency means simply that commands which have the same name will produce the same effect when executed regardless of the operating environment.

2.1.4. THE COMMON COMMUNICATIONS SUPPORT SAA's communications support is its most fully defined element probably because the prior existence of SNA enforced a greater level of consistency among the communications products and implementations. In fact, there are multiple implementations corresponding to most of SNA's several functional layers. Figure 2 lists these implementations, the SAA functional category in which they are grouped and the corresponding SNA transmission layer to which they apply.

2.1.5. THE COMMON APPLICATIONS LIBRARY Common Applications refer not to a product or facility but rather a commitment from IBM that it will develop some applications which conform to the SAA specifications. "These will be key applications that satisfy a significant customer need for use *across the three Systems Application Architecture environments* (emphasis added)."[2]

Allowing for the fact that a list of "key applications" that span all three environments is clearly limited, the Common Applications are currently the most insubstantial portion of the SAA specification.

2.2 WILL SAA CHANGE COMPUTING?

"The announcement is as significant for what it doesn't say as for what it does"

Philip B. Ross, V.P. Research,
Management Science America

"...prominent products...that did not make the SAA list may face tenuous existences"

David Kull, *Computer and Communications Decisions*

"...IBM plans to continue to support the execution of existing applications, thus conserving customers' current investments"

Systems Applications Architecture An Overview

While SAA is obviously still in the early stages of definition, it is equally clear that a siginificant effort has been made to "map" the existing IBM product base into the framework defined by SAA. As a result, some relatively obscure products (e.g. REXX, EZ-VU) are included in the specification while some prominent products are not.

2.2.1. THE COMMON PROGRAMMING INTERFACE. The initial emphasis of the SAA architecture is clearly programming. Since languages are of utmost importance to programmers, it is interesting to analyze the languages that SAA will support in contrast to the ones that have been omitted. COBOL, FORTRAN, and C are all relatively obvious choices for support. COBOL was mandatory because of its prominence as a business applications development language. FORTRAN support was required by engineers who account for a significant percentage of the usage of the installed base of minicomputers. Since portability was central to the SAA "theme", C, liberally supported by compilers for processors of all sizes could not be ignored.

In fact, all three high-level languages have been implemented on a variety of different machines with widely varying architectures. If a common element exists between the three, it is that none of them depend on specific architectural designs or implementations. Notably absent in the SAA specification are otherwise obvious language choices such as IBM's own Assembler and PL/I. These, I contend, were not included in the specification specifically because it is SAA's long-term intent to eliminate architectural dependencies across IBM's entire product line.

The specification of VM's REXX as the basis for the SAA procedures language is probably due to the additional capabilities it offers over MVS' CLIST language and the ISPF Dialogue Manager.

---

1. *Systems Applications Architecture An Overview* GC26-4341, IBM Corporation, p. 2-3.

2. *Ibid.* p 16.

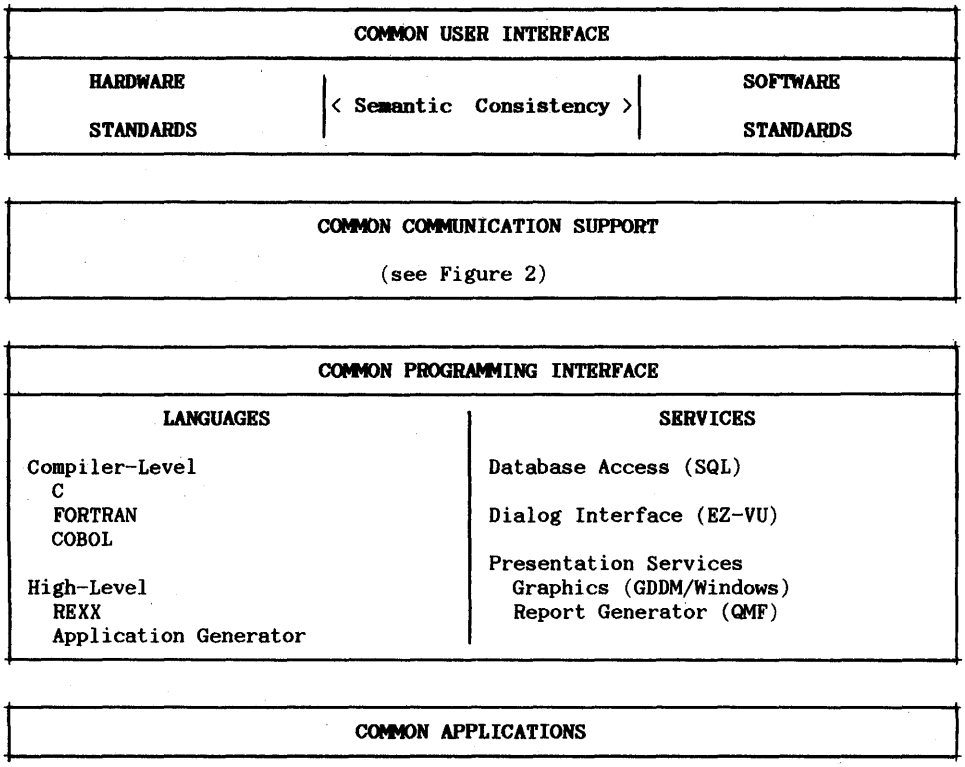


Figure 1. The Components of SAA

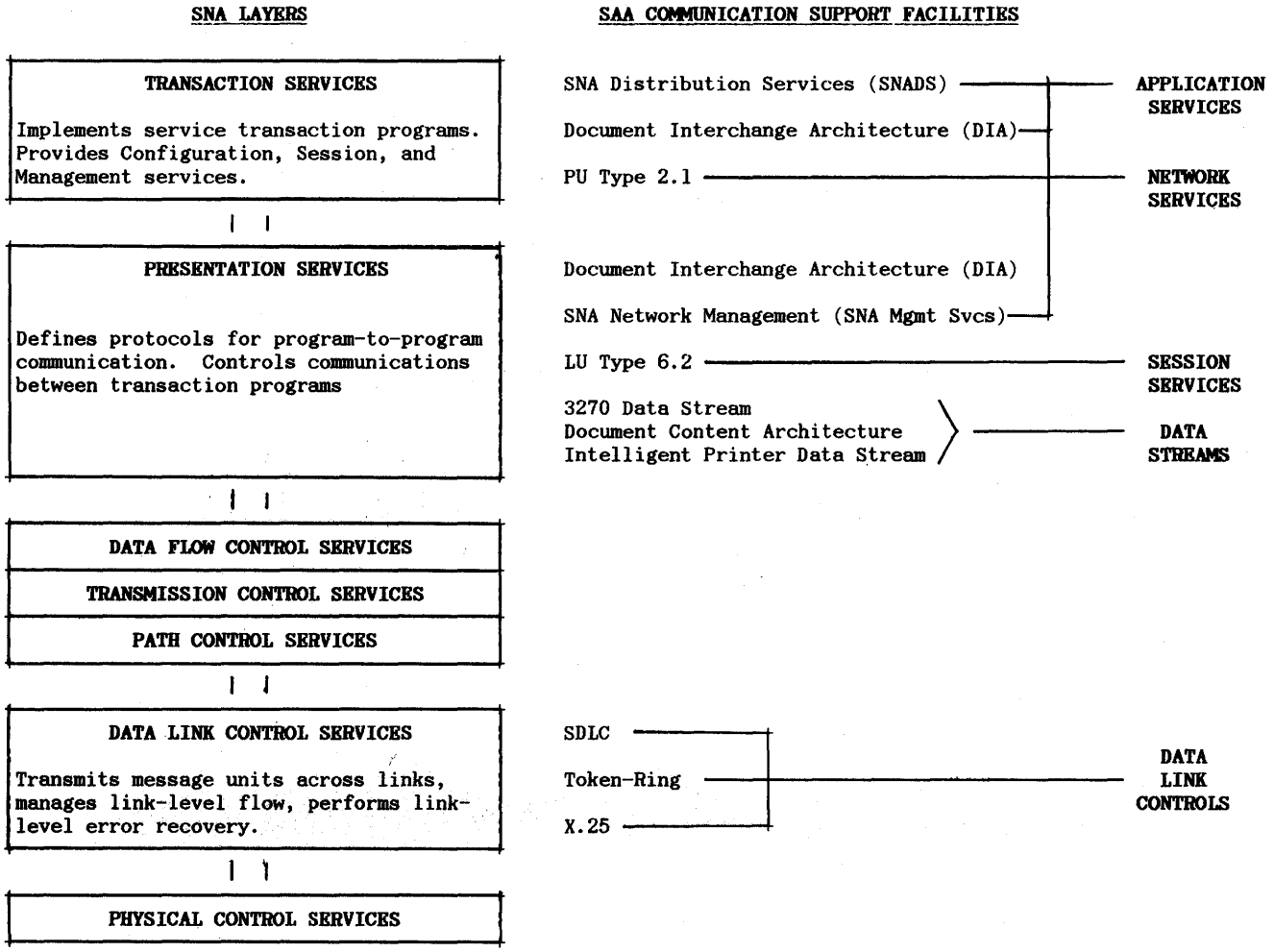


Figure 2. The Relationship of SNA Layers to SAA Communications Support

The Application Generator is the least-defined element in the current specification. While many industry analysts have made much of Cross-System Product being a weak offering, it does indicate that IBM intends to support a split-function generator implementation (i.e. separating the development and execution portions).

Ultimately, these language specifications will impact both in-house developers and vendors. For instance, any shop contemplating starting a major development effort, which might take four years to bring to production and three years of operation to amortize the initial development cost, would have to seriously consider committing to the PL/I language. Similarly, long term commitments to vendor products written in PL/I or Assembler will have to be reevaluated. How soon the SAA "blackball" will begin to be felt remains to be seen. I suspect that it will "trickle up" from the PC world to the mainframe and predict that in another year "SAA Compatability" will be a selling point for some major PC products.

2.2.2. THE COMMON COMMUNICATIONS SUPPORT. The communications suport focuses on program-to-program communication and low-level (PU 2.1) networking. These elements are key to integrating intelligent devices into SNA networks but their potential, if realized, would significantly overshadow existing forms of data communication. In short, as the following statements show, 3270-type terminals are becoming obsolete.

"The high bandwidth of the processor-to-keyboard/display of such systems as the IBM PC cannot be provided with a terminal such as the 327X."

G. Glenn Henry, Manager of Hardware and Software System Development for the IBM PC/RT

"...SNA will continue to exploit advances in technology as they appear. Here, a known requirement is to extend the peer-to-peer operation in SNA. This need follows from developments both in processor design--especially in small systems--and in transmission technology....

Robert J. Sundstrom, Phd. IBM's Manager of Communication Systems Architecture

"OS/2 is going to be...the terminal of choice in host attachments...."

James B. Archer, IBM Systems Manager, Operating Systems and Extensions

More importantly, the applications subsystems designed to support them, specifically CICS and IMS/DC, will also be obsoleted by LU 6.2 functionality. What user would willingly lock himself into a CICS application if he could have windows, pop-up menus, multiple session, graphics, etc., and probably faster response to boot?

3. WHAT WILL THE SAA ENVIRONMENT LOOK LIKE?

"In some years we may think of all of the computing power at a single...site of business activity...as effectively a single node, regardless of its physical disbursement...and...number of 'local operating systems'"

Harold Lorin, Consulting Faculty Member, IBM Systems Research Institute

Most analysts see SAA as an attempt to provide continuity between IBM's many diverse mid-range architectures and its mainframe and microcomputer environments. They imply that such diversity is at best a liability and, at worst, a potentially disabling strategic and tactical blunder.

However, IBM maintains that it:

> ...offers systems based on several different hardware architectures and control programs....[in order] to provide products with outstanding price/performance ratios across an ever-broadening spectrum of customer requirements....[which spans] a nearly thousand-fold capacity range....[3]

This approach makes good sense if these "non-compatible" systems are used in a stand-alone manner or only in concert with each other. And there is reasonable evidence to indicate that this has been the case in the past. No definitive figures are available on the percentage of "departmental-class" minicomputers installed which are attached, in some way, to a mainframe computer, but it is widely reported that IBM expects only 10 to 15 percent of the 9370s to be. Industry analyst Rick Martin of Sanford C. Bernstein & Co., member of the New York Stock Exchange, says "Only about one percent of Digital's installed base communicates with IBM mainframes...."[4]

3.1. DRIVING FORCES

If IBM has found it previously uneconomic to engineer compatability across its product lines, it seems only reasonable that some change has encouraged the SAA effort now. In fact, these "driving forces" have been discussed in several articles in the *IBM Systems Journal.*

Dr. Alan Scherr, an IBM fellow, claims there are strong technical reasons for a tiered approach to information processing. Dr. Scherr sees departmental computing as a logical extension of the distributed computing concepts of the mid-to-late 1970s[5]. He notes that historical trends in configuring systems in networks have resulted from trade-offs based on two factors:

- o the largest systems had the best price/performance ratios
- o the cost of computing was relatively high compared to the cost of communications

This has resulted in four basic configurations of networks being used, all of which are asymetric (i.e. computer-to-terminal) in nature. These configurations are shown in Figure 5.

Price considerations are transitory in nature, changing from one class of machine to another with technological innovation and maturity. However, Scherr claims, there are currently "underlying technology-based pressures that...motivate the design of complex networks of systems."[6] These are:

- o A larger range of available capacity with greater overlap between size-classes
- o No discernable economies of scale between size-classes
- o Communications costs greater relative to computing costs

---

3. IBM Corporation, *Systems Application Architecture An Overview,* GC-26-4341-0, May 26, 1987.

4. Freedman, David H., "digital vs. IBM", *Infosystems,* May, 1987 p. 46.

5. Scherr, A.L.,"Structures for Networks of Systems", *IBM Systems Journal,* Vol. 26, No. 1, 1987, p. 4.

6. *Ibid.*

The range of available capacity gives greater freedom in the selection of a class of machine to address a requirement. Where certain types of database applications might have required a mainframe computer a decade ago, today this requirement might easily be filled by a mini or even a large microcomputer. This cpapcity range applies not only to processor power (MIPS), but to memory and disk storage as well. Inexpensive microcomputers are now available with four megabytes of memory and over two hundred megabytes of disk storage.

End-users tend to think of computing economy in terms of price/performance ratios. This is too simplistic a criteria for assessing the departmental or personal environments. Productivity is a function of the amount of work done versus the total cost to perform the work. Large machines have an advantage when the workloads are large because of their richer hardware and software architectures and because of their typically greater duty cycle. In contrast, smaller machines benefit from reduced costs due to lower software overhead and a simpler structure. Scherr estimates that these factors are roughly equal given today's technology, thus the cost per MIP is approximately equal across mainframe, minicomputers and microcomputers.

Communications costs will not decrease (at least for the foreseeable future) as rapidly as computing costs. Also, sustaining communications absorbs a significant amount of computer power. Scherr estimates that the number of instructions required to retrieve a disk record across a communications link is five times that required to access a local disk. Therefore, placing computing capacity where it avoids communications is an increasingly attractive trade-off.

3.2. LOCATION OF FUNCTION IN A PROCESSOR NETWORK

The location of function in a processor network results from resolving a number of trade-off decisions which may be different for each installation and application. These trade-offs generally involve compromises between ease of use, performance, data access and availablilty. Location has generally implied a class/size of processor, i.e. centralized computing is usually done on a mainframe; however, this is not strictly a requirement. Therefore, this paper will refer to the "direction" of function placement, with *upstream* meaning more centralized and *downstream* meaning more distributed.

3.2.1. EASE OF USE. Small systems have typically offered a richer user interface than large systems. This originally resulted from the requirement for these systems to be useable by a single-user, which implied a consistent, easily learned and understood, "total system" interface supporting installation, operation, programming, and maintenance. The implementation of this interface was made feasible by a high-bandwidth path between the processor and the "console". Such interfaces have set a standard for user interfaces (e.g. windowing, keystroke-by-keystroke responses, and graphics) which are impractical to implement in a terminal (e.g. 327X) environment. Therefore, the optimal place to implement the interface is within the workstation itself.

3.2.2. PERFORMANCE. Performance, except in the case of extremely computation-dependent processes, is largely maximized by minimizing communications requirements. To the greatest extent possible, data should be stored in the node where the application program using it will be executed, thus reducing communication delay.

In addition, the effect of advanced technology is to make incremental increases in "raw" processor power relatively cheaper in small machines. As G. Glenn Henry points out[7]:

> [The] influence of VLSI is a reduction in the fraction of the cost of a computer that is represented by the cost of its logic components....[As a result] the cost of small systems [is] rapidly approaching an asymptote formed by the cost of their frames, covers, power supplies, and manufacturing costs...."

In other words, small systems can utilize extremely advanced logic design as long as they can be physically packaged in relatively small units. Prime examples of this effect are the PC/RT and PS/2 systems which are cheaper than their predecessors in spite of their more elaborate logic designs. As a result, technology has begun to favor smaller rather than larger machines for simply computational workloads.

3.2.3. DATA ACCESS AND MANAGEMENT. Unfortunately, it is frequently not possible to minimize communications simply by placing sufficient processor power locally. Few "front office" applications are isolated to an individual small work-group. Even applications which are highly localized (e.g. branch-office banking, retailing, etc.) require remote access to data to support the "foreign" customer. This implies that either peer-to-peer access to information stored on separate distributed nodes is supported or that a centralized "master file" is maintained.

Data management (synchronization, security and integrity) requirements are independent of data location (upstream or downstream). However, data control is more expensive the smaller the volume of data managed becomes. Thus the liability of less rigorous data managment (or the additional data management cost) resulting from moving data downstream must be weighed against increased communications costs.

3.2.4. AVAILABILITY. Availability is also a major consideration in locating function. From an overall enterprise standpoint, *horizontal distribution* (implementing the same functions in independent peer nodes) results in increased percentage availability[8]; however, the availability to any single workgroup remains the same. Therefore, functions requiring one hundred percent availability may be more effectively implemented upstream where expensive measures (e.g. redundant hardware, fault-tolerant processors, etc.) can be cost-effectively implemented. Note that as logic gets cheaper, it will become more feasible to implement hardware (tightly-coupled) fault-tolerance (a la System 88/Stratus) in smaller processors.

Practically, in current implementations, communications is much less reliable than computing and many more outages are caused by software failures than by hardware failures. Therefore, relatively stable downstream processors are likely to be more reliable than the more volatle upstream processors. On the other hand, upstream processors are more likely to have a larger support staff, thus reducing the mean-time-to-repair (MTTR) a failure.

3.2.5. ACCESS TO OUTPUT FACILITIES. As the user interface becomes more sophisticated, so does the requirement for improved hard-copy output. This creates a demand for expensive output devices such as: laser printers, plotters, slide-makers, etc. These devices

---

7. "IBM Small System Architecture and Design - Past, Present, and Future", *IBM Systems Journal,* Vol 25, Nos. 3/4 1986, p. 325.

8. Increasing the number of nodes decreases the probability that all nodes will be down, assuming the individual reliability of each node is the same. However, it also decreases the probability that all nodes will be up.

are too costly to be attached to an individual workstation, so an additional functional requirement, the ability to share output devices, is created.

3.3. STRUCTURE OF AN SAA COMPUTING ENVIRONMENT

The resolution of these trade-offs implies a four-tier hierarchy (which is totally compatible with the three SAA environments) based on isolation and distribution of data and function in such a way as to minimize the amount of data transfer required to process the majority of the workload. The four tiers:

- o Centralized
- o Enterprise (Site)
- o Departmental
- o Workstation (Private)

are shown in Figure 3.

The end-user accesses the processor network via a workstation which may be a "dumb" (e.g. 3270) terminal but is most likely a personal computer. The workstation stores individual data such as: draft documents, spreadsheets, individual telephone lists, "tickler files", personal schedules, etc. It also stores the programs required to manipulate that data and to support the interface(s) to upstream nodes in the network. These interfaces may be via LAN, APPC (LU6.2), 3270 protocol, DECNET, etc.

The small work-group or *departmental* processor is accessible to all workstations within the group and allows them to share data. This processor also provides communications to the other levels in the hierarchy. Applications related to the primary function of the work-group execute on this processor and data utilized by these applications are stored on this processor. Many of these functions involve external interfaces, such as: shipping, receiving, order entry, accounts payable, accounts receivable, etc. Also supported are generic functions such as: the department budget, document storage, time accounting, information logging, etc.

If no *departmental* applications are run by the work-group, this function can be supported by *servers* (typically a LAN) which provide storage and output device sharing. This is particularly significant (at least in the near term), because the majority of work-groups are not organized so as to minimize the sharing of data with other work-groups. Thus, the previously defined criteria encourages running applications upstream of the departmental level.

The departmental processors at a given site attach to an *establishment* processor. This may be one of the enterprise's "backbone" mainframes or may be another departmental-size processor. This node supports functions which are unique to the entire site such as communications concentration/gateway facilities to the enterprise's wide-area network.

At the upstream "end" of the network is the centralized processor. This is the traditional mainframe location and will typically support the back-office applications and enterprise-wide database support.

3.4. Future Evolution of SAA

The near-term evolution of SAA will certainly change the nature of IBM-compatibile computing. Instead of a highly centralized computing resource relying on large mainframes, the SAA environment might be called an *extended data center* in which processing is largely done remotely and mainframes are utilized for data management synchronization and backup, wide-area network management, and the support of highly centralized applications.

While many of the facilities required in the SAA environment are currently implemented in existing products, what is still lacking is adequate database support. SAA will require a distributed database architecture which can operate across multiple processors and processor architectures and still provide a single database image. This image would include file and record locking/update in a manner transparent to the user. Attaining these capabilities with sufficient performance may bring on SAA's second phase, the replacement of the System/370 architecture.

4. SAA, THE BROADER VIEW

"[A system's architecture is a hierarchy made up of levels. In] a perfectly encapsulated hierarchy...the nature of any level...would be of no concern [except to the level directly above it.] In contemporary systems structure, however, multiple lower levels are visible to a higher-level function."

"The full range of compatibility, portability, and cohabitability issues of current computing systems, hardware and software, derive from the multilayered views that are available to end users, application programs, subsystems, etc."

Harold Lorin, Consulting Faculty Member,
IBM Systems Research Institute

SAA's stated goal is to facilitate portability between dissimilar architectural environments. Few, if any, industry analysts have considered the ultimate implications of this effort. Once the SAA interfaces are in place and the Common Application Library has been populated, it will be possible to migrate mainframe computing beyond the constraints of the System/370 architecture.

Conceived in the late 50's and early 60's, the System/360 architecture preceeds most of the major forces in computing today, such as:

- o Virtual Storage
- o Microprocessors
- o Database Technology
- o Data Communications
- o High Software Cost versus Low Hardware Cost

While virtual storage was later implemented in the System/360 architecture (initially in the Model 67 and with caching in the Model 85), the major contribution of the System/370 evolution was to provide virtual storage as a standard feature of the product line. The recent XA extensions to the System/370 architecture were designed to relieve virtual storage constraints which had developed because of limitations in the original implementation. This section discusses some of the key architectural and design decisions which were made in the System/360 - System/370 implementations and how those decisions have begun to restrict mainframe computing capabilities. With that base, we will discuss various projects and products which foreshadow how future SAA-compatible processors will avoid these constraints.

4.1. THE SYSTEM/370 ARCHITECTURE

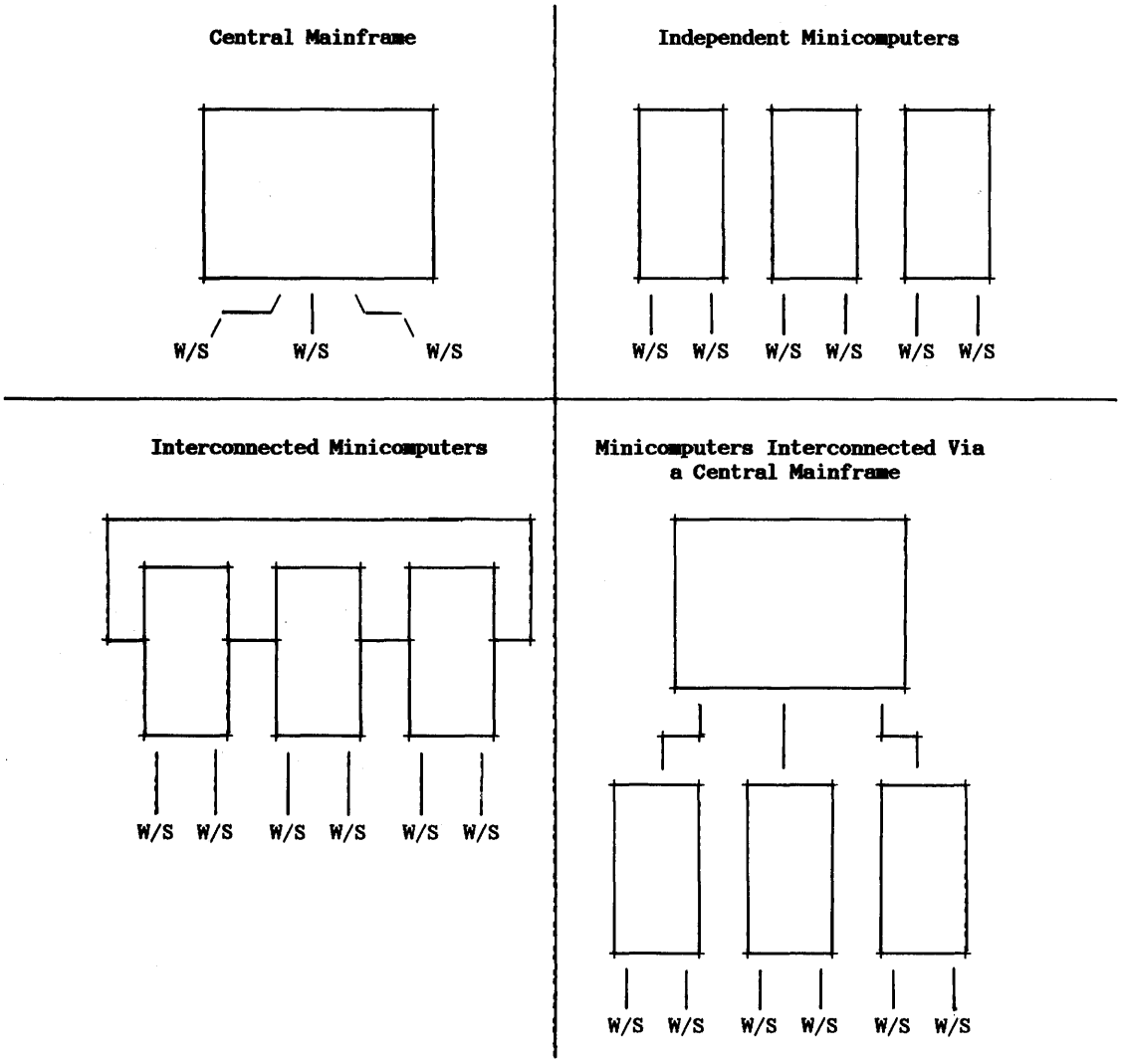


Figure 3. Pre-SAA Network Configurations

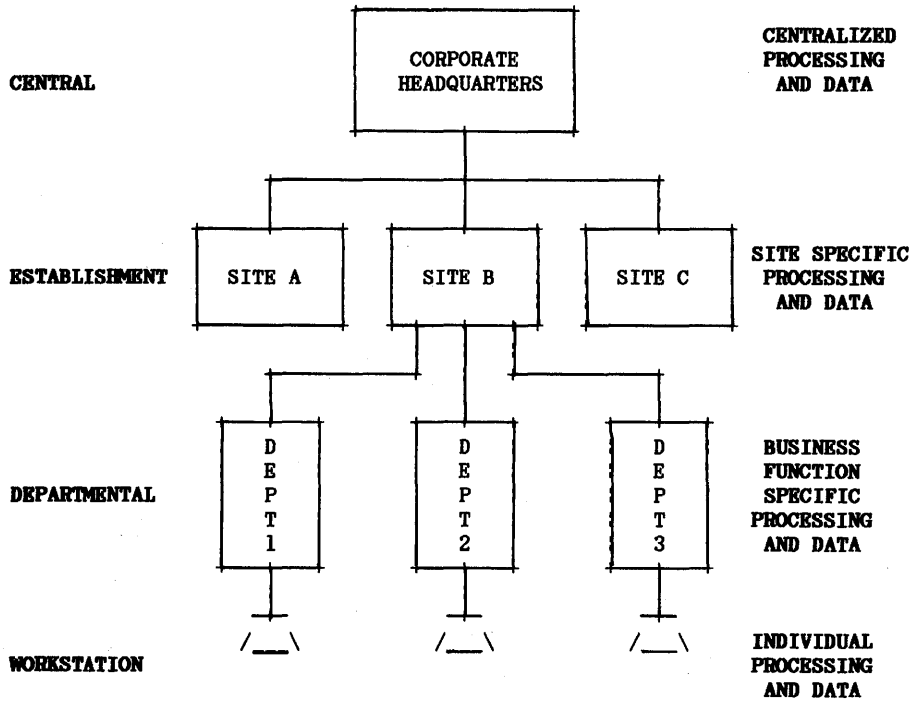


Figure 4. The SAA Computing Hierarchy

"Because they lacked clear criteria, computer architects between 1946 and 1975 tended to specify ever more complex systems....The computers acquired more instructions, more addressing modes, more data types, and more special features."

"The technology to implement a machine with many diverse instructions was provided [in the System/360] by very fast read-only storage that permitted the economical implementation of microcode. This ushered in a new era of machines with a great diversity of instructions."

Martin E. Hopkins, Manager of Compilers,
IBM Advanced Minicomputer Department

The major components of the System/360 implementation are: device-independent I/O and microcoded instructions. The device independent I/O channel architecture resulted from the recognition that I/O devices were proliferating and that supporting them had become a programming burden. The S/360 channel architecture isolated the details of specific devices from the CPU and thus from the programmer. The use of microcoded instructions allowed a large, complex instruction set to be implemented across a wide range of processor sizes; this instruction set contained many specialized instructions which were supposed to simplify programming in specific environments. Thus, the S/360 is very much a programmer's machine, specifically an Assembler-language programmer's machine.

The refinement introduced by the System/370 was cached virtual memory. Virtual memory reflected the programmer's need for larger address spaces because of the complexity of overlay coding. The cache was simply a way to increase the performance of virtual storage.

"In 1975...most System/370 machines were implemented with a cache and microstore....The cache had been introduced to provide faster access to main storage....The CPU fetched instructions from the cache and interpreted them on the basis of a microprogram in the microstore....Neither the cache nor the microstore is part of System/370 architecture."

Martin E. Hopkins, Manager of Compilers,
IBM Advanced Minicomputer Department

The System/370 hardware and operating system implementation is relatively tightly-coupled. A significant portion of the operating system code supports memory management and the implementation of that function is both pervasive and restrictive.

"In software architecture, the concepts of pages and segments as well as the use of segment and page tables for translation are so thoroughly entrenched that they are impossible to change. Early virtual memory designers decided to use tables for translation [because it was argued] that the use of software rather than strictly hardware to do the translation would not lock the architecture....In retrospect, there would be more flexibility in a hardware implementation. With the ever-increasing cost and complexity of system programming and the decreasing cost of VLSI, it is becoming more attractive in many cases to implement functions in hardware rather than software."

Richard E. Matick, Phd.,
Thomas J. Watson Research Center

Twenty-plus years of experience with the System/360 - System/370 architecture has made several things clear:

- o It is very difficult to identify instruction sets that are specific to a particular programming environment.
- o Implementations of functions in hardware may not be more restrictive than software implementations.
- o Tight coupling between the operating system and the underlying hardware tends to restrict the evolution of both.

Based on these observations, SAA can be viewed, in a larger sense, as an attempt to "decouple" various aspects of overall system architecture. We can expect this to be accomplished by more clearly defining the interfaces and separating functions. This effort, over time, will completely change mainframe design.

Key components of the new SAA-compatible architecture are already under development and, in some cases, are partially implemented in various products. The essentials of these efforts are discussed in the following sections.

4.2. PROCESSOR ARCHITECTURE

"A 'good architecture' (processor architecture) should be easily and efficiently represented in the technology, and should provide for simple, efficient compilation.

Harold Lorin, Consulting Faculty Member,
IBM Systems Research Institute

"In 1975 the 801 project was started at...IBM....to reexamine the trend toward complexity. The goal was to discover how to deliver the most computing power at the lowest cost in an environment geared to programmer productivity."

"This architectural point of view has become known as RISC....[which] nicely captures the spirit of research into the premise that less is better."

Martin E. Hopkins, Manager of Compilers,
IBM Advanced Minicomputer Department

Each successive new processor design introduced by IBM has offered increased CPU performance (MIPS) for the same price. Typically, the performance increase has been by approximately a factor of two. A doubling of processor speed has offered sufficient incentive for the installed base to upgrade without "over-powering" the marketplace to the point that future sales are endangered.

Traditionally, this increase in power has been arrived at in one of two ways:

- o The processor is implemented with faster logic
- o The design is made increasingly complex via increased use of pipelining, etc.

Recently, these techniques have been used alternately to increment processor speed. Thus 3033's were designed with elaborate pipelines but relatively slow logic, 3084's were much simpler in design but employed faster logic, and 3090's implement more sophisticated piplines without an increase in circuit speed over the 3084. Each approach involves a trade-off in the length of time required to "perfect" a design and its subsequent reliability in the field.

With the TCM package design (which is the basis for all the above-referenced processors) completed, it became fairly obvious that a maximum reasonable complexity level had been reached and that logic speeds could not be increased indefinitely. This prompted research into alternatives like the 801, RISC-architecture project.

The concept of a RISC (Reduced Instruction-Set Computer) processor is simple. Instruction traces originally performed as part of the research into the caching algorithm for the System/360 Model 85 had indicated that ten assembler instruction accounted for two-thirds of all instructions executed. A RISC machine is designed to optimize the most heavily used instructions and to eliminate from the instruction set those functions which can be accomplished by combinations of simpler instructions.

All instructions in the 801 instruction set can be executed in one cycle. Since the standard System/370 instruction requires eight cycles, this offers a basic increase of nearly an order of magnitude for those portions of the two instruction set which overlap.

Further refinements of the 801 design include thirty-two registers rather than the System/370's sixteen and the replacement of the S/370 microstore with an instruction cache separate from the data cache. More registers means potentially less overhead on branches and the instruction cache allows major routines to be executed at S/370 microcode speed. Thus, the 801 processor frequently offers a twenty-to-one improvement in execution speed over the S/370 with a penalty of an approximate twenty-percent increase in the number of instructions executed.

The advantage of processors based on the 801 concepts are the potential for three-to-four processor generations' "worth" of speed improvements realized without new circuit technology and with reduced complexity in the design.

It is significant to note, that the 801's processor design is near the complexity level of the current 3090 I-unit. This reduced complexity is extremely significant for new product development because it reduces development lead-time and cost. Both time and cost were critical considerations in the original design of the TCM and I suspect any significantly less-complex design which did not reduce performance would be seriously considered. For these reasons, I expect processor design for the SAA environment to borrow heavily from the 801 design.

4.3 OPERATING SYSTEMS DESIGN

"The designers [of the PC/RT]...wanted to achieve a higher degree of program isolation from the hardware...."

"[This is accomplished by using] a Virtual Resource Manager (VRM) to control the real hardware of the machine. The VRM presents operating systems with a Virtual Machine Interface (VMI) that not only conceals the complexities of [the design] but provides the guest operating system with a significantly more powerful set of functions than are available on the bare machine."

"[It is this] system architecture that permits accelerated adoption of new technological advances."

G. Glenn Henry, Manager of Hardware and Software System Development for the IBM PC/RT

"Shall we integrate new function into older operating systems, or should we begin to "hide" these older systems, to think of them only as components of a single operating system that contains all software elements and maintain them only so long as we maintain the architectures on which they run?"

Harold Lorin, Consulting Faculty Member, IBM Systems Research Institute

It has often been rumored, that MVS would become more VM-like (or even that some implementation of VM would replace MVS). Given that SAA implies "decoupling" of architectural components, I would expect that these rumors have a kernel (but only a small one) of truth.

If, indeed, an entirely new processor architecture is to be implemented, then MVS as we know it will be obsolete. Further, I would expect this new processor family to be designed so that it is relatively operating system independent. This could be accomplished by utilizing a facility analogous to the Virtual Resource Manager implemented in the current PC/RT.

The Virtual Resource Manager is essentially an emulator which provides the functions of a well-defined but physically nonexistant virtual machine. The operating system interfaces with this virtual machine rather than with the hardware itself. This design allows for multiple guest operating systems like VM does (which would be mandatory for transitions to a new operating system) but can be implemented without the VM performance penalty.

Use of a Virtual Machine Interface would free both the hardware and the operating system designers from dependence on one another. With the speed at which new hardware developments are being implemented, an implementation which does not provide this separation may well block the logical evolution of hardware (or software) design. In addition, this implementation is well suited to the support of special-purpose hardware such as: application specific processors, associative memory, etc., which would allow tailoring a system to an installation's needs at a much higher level than has been previously possible. For these reasons, I expect operating system design for the SAA environment to be implemented similar to the PC/RT's Virtual Resource Manager.

4.4. MEMORY MANAGEMENT

"...controlling and managing the virtual memory hierarchy is a pervasive task requiring substantial operating-system resources, as well as hardware and architectural assistance. If a very large, fast memory could be implemented to replace the memory hierarchy, substantial simplification in hardware and software could result.

Richard E. Matick, Phd.,
Thomas J. Watson Research Center

"In the future, the largest improvements in performance will probably come about through improving the memory hierarchy."

Martin E. Hopkins, Manager of Compilers,
IBM Advanced Minicomputer Department

"...a machine architecture with massive amounts of [non-volatile] memory...for use as an in-memory database....could do away with resource recovery in the database architecture....[and] provide extremely fast service....The implications of such an architecture are currently under study."

Patricia G. Sellinger, Phd.
Manager of IBM's Database Technology Institute

The design of the memory hierarchy in System/370 reflects a time when memory was relatively expensive. Since the quantity of real memory that could be installed was limited, virtual memory was implemented as a compromise to avoid constraining software development. Essentially, paging can be viewed as a way of standardizing and simplifying overlay coding. Thus, its goal was to reduce programmer effort.

Subsequent developments have dramatically reduced the price of memory and require the evaluation of new approaches to the management and usage of memory. Three trends seem evident:

- o Implementation of extremely large memories is feasible
- o Separation of memory function into instruction and data memory
- o Development of special-purpose memories with built-in associative search capabilities to support database requirements

The first two trends are currently being implemented in a rudimentary way within the existing System/370 architecture. The new 3990 DASD Subsystem supports cache sizes up to 256 megabytes. Even a moderately sized DASD pool implemented with these large cache sizes would constitute a very large "total system memory" which is optimized for data rather than instructions. If the path to these DASD caches had sufficient bandwidth, the distinction between DASD cache and main memory would be further blurred.

Further in the future, is associative memory. This special purpose memory would be capable of doing the compares required by database searches without using the CPU. This is an example of special-purpose hardware designed to support a specific (although pervasive) function.

Regardless of what specific facilities are implemented, we can expect the memory design for the SAA environment to be much less closely associated with the design of the processor. Certainly it will utilize very large memory sizes which may be segmented for specific purposes. Very likely, the control of memory will require a larger portion of the total system's logic than the CPU, just as it currently does in the PC/RT.

4.5. I/O DESIGN

"[The] speed differential between memory and input/output devices...has been the main factor influencing I/O architecture."

"A significant problem with large, fast computers is that of providing sufficient I/O bandwidth to keep the CPU busy....[As a result] each succeeding generation of computers has increasing numbers of channels and I/O devices....If the I/O devices ever become very much faster, many fewer will be required to keep the CPU busy."

Richard E. Matick, Phd.,
Thomas J. Watson Research Center

As processor speeds have increased, the requirement to supply them with data has increased. However, since device transfer rates have not kept pace, this requirement has been satisfied by adding more channels and devices. Clearly, this approach attacks the symptoms rather than the cause of the problem.

Two practical methods of addressing this problem are emerging: deep buffering at multiple levels within the I/O process (mentioned in the preceeding section) and more evenly splitting the I/O workload between the I/O subsystem and the CPU. It is only a small conceptual leap from the current channel architecture to one in which I/O requests are directed to an "I/O processor" which obtains the record and places it into main memory without further intervention from the CPU. This approach would build logically on the currently implemented large cache buffering approach.

4.6. LANGUAGES

"It should be kept clearly in mind that the term *software* really means the use of *general-purpose* hardware rather than special-purpose hardware."

Richard E. Matick, Phd.,
Thomas J. Watson Research Center

"Programming has long been recognized as a bottleneck....because today most programming is done in high-level language....the exact nature of the computer should concern only the compiler writers...."

Martin E. Hopkins, Manager of Compilers,
IBM Advanced Minicomputer Department

"The microcode approach arose in IBM because it seemed necessary to produce....a compatible family of machines with a wide range of cost, [utilizing] the same sort of instructions. Increased reliance on high-level language programming tended to undercut the argument for high-level operations based on ease of programming."

Martin E. Hopkins, Manager of Compilers,
IBM Advanced Minicomputer Department

If System/370 was the assembler programmer's architecture, then SAA will be the application programmer's architecture. The emphasis in SAA will be on languages which are not dependent on the underlying architecture of the processor. For this reason, an assembler for a "true" SAA processor may never be commercially released.

SAA processors will be optimized for compilers. In fact, one implication of the 801 implementation was that manually writing efficient code was non-productive because reordering, linkage conventions, etc., were simply too complex for human beings.

However, I doubt that SAA will endorse HLL (high-level language) implementations in the forseeable future simply because the advantages are too abstract to be measured. Therefore, I don't believe the professional programmer is jeopardized by SAA.

4.7. DATABASES

"The data base of an organization is one of its most significant assets....The existance of increasing amounts of parochially held data is a threat to the efficiency of the organization."

"...it will be necessary to devise ways of making the growing body of data held on small systems available to the remainder of the organization."

G. Glenn Henry, Manager of Hardware and Software
System Development for the IBM PC/RT

"...microprocessors [make] it economically feasible to have computer architectures with intelligent devices such as intelligent disk controllers, database search engines, and special-purpose dedicated machines, e.g., for message processing....One of the key problems is to invent a solution that will not prove obsolete as outboard microprocessors, memory devices, and communications become even faster and cheaper.

Patricia G. Sellinger, Phd.
Manager of IBM's Database Technology Institute

The management, coordination, access control, synchronization, etc., of multiple distributed databases in the SAA environment is going to be a significant task. If I were to guess at an area in which the SAA architecture was most likely to be ultimately self-limiting (as the 370 architecture has been in terms of virtual

storage management), this would be it.

Small systems (e.g. S/38) have recognized the need for an integrated database for some time. I anticipate that SAA will percolate this requirement up to the mainframe and that this database will enjoy significant design and implementation hardware assistance if not actual architectural assistance.

IBM has already stated that its direction lies with DB2. Effective hardware assistance could easily allow DB2 to surpass IMS' (or other hierarchical databases) performance level, especially since that hardware assistance could not be shared by IMS. For this reason, I expect functional enhancements to IMS to taper off in the near future.

4.8. DATA COMMUNICATIONS

"The underlying phenomena that motivates the need for more uniform software interfaces and for more function sharing across units on a network is the anticipation of dramatic improvement in interconnect facility. One provocative speculation comes from the convergence of I/O and local-area network technologies."

Harold Lorin, Consulting Faculty Member, IBM Systems Research Institute

"With the steadily decreasing cost of mini- and micro-processors and storage, more and more of the peripheral nodes are small *systems* such as personal computers, distributed processors, intelligent workstations, and office systems.

"A number of LU types are defined in SNA, with the earlier ones (LU types 1, 2, and 3) optimized for asymetric host-application-to-device communication."

Robert J. Sundstrom, Phd. IBM's Manager of Communication Systems Architecture

"The high bandwidth of the processor-to-keyboard/display of such systems as the IBM PC cannot be provided with a terminal such as the 327X."

G. Glenn Henry, Manager of Hardware and Software System Development for the IBM PC/RT

The requirements of the SAA environment are pushing SNA vigorously toward peer-to-peer communication. SAA will obsolete conventional, asymetric (host-to-terminal) communication simply because it cannot offer the level of features. Specifically, this means that new implementations based on IMS/DC, CICS, etc., are poor investments. More than any other aspect of SAA, this effect will sorely try vendors and end-users alike.

Longer term, SAA will blur the distinction between I/O and data communications. Given that an I/O processor design similar to that discussed previously is implemented and that wide-area networks can be implemented with very high bandwidths, what is the difference between a request for a record from a database on another processor and a "local" I/O?

5. MANAGING THE SAA ENVIRONMENT

Previously we have discussed the effects SAA's layered architecture and emphasis on applications programming will have on the design of various hardware and software components. In this section we will concentrate on the changes that SAA will have on the traditional IS management functions.

Managing the SAA environment presents significant additional challenges to the performance analyst, capacity planner, database administrator, space manager, operations manager, etc. Most of these problems are not actually unique to the SAA environment but are magnified by it either because the site of the problem is physically removed from the management group or because the number of sites in which the problem can occur has proliferated.

The functional effect of SAA's layered architecture is to move many key business processes outboard of the mainframe environment into departmental "servers" as shown in Figure 3. These "servers" may span a gamut of hardware environments, from LAN servers through supermicros to various sizes of minicomputers, and software environments, including MVS, VM, OS/2, non-IBM minicomputer operating systems, and eventually post-MVS mainframe operating systems.

Some analysts have implied that traditional I/S management disiplines such as: performance analysis, service level management, capacity planning, accounting and chargeback, etc., are not important in departmental computing environments because the installed hardware base is so much less expensive. However, an enterprise operating several hundred departmental computers in addition to the mainframes, may actually expend the majority of its total I/S dollars outside the central data center. The fact that those dollars come out of many different budgets and that the individual purchases are smaller does not diminish the total amount the enterprise has "at risk".

Except at the establishment and some mainframe site nodes, these processors will be locally supported by a far less technically capable staff than currently supports mainframe installations. This places considerable additional burden on the I/S technical management personnel. Some of these problem are outlined in the following sections.

5.1. PERFORMANCE ANALYSIS

Performance is a major consideration for adopting a departmental computing configuration. In general, this performance is obtained by minimizing the number of communication delays. In some cases, the performance analyst will be involved in database design decisions. To maximize cost/performance the optimum split between database content at the departmental level, the number of upstream accesses, and cost of synchronizing the establishment and site databases with the departmental database, will have to be weighed.

Accesses to upstream processors will also have to be optimized and subsequent upstream service tracked. This will typically require response time as seen by the downstream node by transaction type. Downstream response time measures will have to be integrated with upstream processor measures to determine the amount of time spent in the upstream processor versus the communication delay.

5.2. CAPACITY PLANNING

One of the significant advantages of a departmental computing implementation is the finer *granularity* of capacity increments available. Because capacity can be purchased, when required, in smaller units, the potential exists for the enterprise to operate with much less average excess capacity. This increases the cost effectiveness of I/S because more of the installed resources are actually being utilized.

Because the capacity increments will be less expensive, less effort will be required to cost justify them. As a result, a natural temptation will be to attack the majority of performance problems with additional capacity. This could result in a far greater percentage excess capacity being installed at departmental sites than would be tolerated in a mainframe installation, thus turning a cost saving opportunity into an increased

expenditure.

Each capacity upgrade will typically require an assessment of upcoming capacity requirements throughout the entire processor "network". In many cases, capacity upgrades will not be made by addition but by substitution. If the replaced processor is owned by the enterprise, a decision concerning its disposal will have to be made. If another node will soon require an upgrade and the previously replaced processor has sufficient capacity a second substitution might be made. This "ripple effect" would proceed throughout the "network" until the enterprise is left with the least expensive processor (smallest, slowest, oldest, etc.) to dispose of to an outside entity.

Given the potential proliferation of hardware and software architectures which could exist, some processors will be replaced by others of compatible, but entirely different design. Correctly sizing these dissimilar machines will require modeling and simulation tools which account for their architectural differences. Thus SAA demands a new, generic view of processor configurations as well as the conventional model-specific view.

5.3. RELIABILITY/AVAILABILITY MANAGEMENT

Enhanced reliability and availability is a further reason to adopt a departmental computing strategy. However, as was pointed out previously, multiple processors increase percentage availability without necessarily improving the reliability of the individual processor. Installations which distribute functions over many small processors can anticipate that one or more of them may be inoperable at any time.

Since few of these small-processor locations will have redundant hardware, potentially long outages will probably be addressed by moving the critical applications to the site's mainframe processor. This is approximately equivalent to offsite backup in today's more centralized environment.

Minimizing the effects of these outages will require extensive incident tracking and correlation of conditions related to outages. Changes which cause instability must be backed off of the hardware/software configurations which they compromise. Similarly, fixes and/or circumventions must be migrated onto these configurations as rapidly as possible.

5.4. INVENTORY/CONFIGURATION MANAGEMENT

Inventory and configuration management is closely related to the reliability/availability management problem. Currently, installations find it burdensome to maintain an accurate hardware inventory for medium-to-large networks. In the SAA environment, this inventory is further complicated by the addition of various processor types and multiple versions of operating system and application software at remote locations. Managing this complexity will require considerably more robust inventory management tools than exist today.

For example, a given departmental processor might support three "front-office" applications in addition to three office automation applications (e.g. word processing, electronic mail, personal calendar, etc.). Each of these applications might exist at one of three maintenance/enhancement levels, old version, current version, or new version. In addition, these applications might be supported under three different operating systems: MVS/IS, VM/IS, and a non-IBM operating system such as VMS. Each operating system environment might be supported on two different processor models and at three different release levels analogous to the application levels. This results in potentially 189 distinct software environments. If the enterprise operates 100 departmental installations, each of which have different hardware configurations, the overall installation complexity is dramatically increased.

5.5. DATA MANAGEMENT

Data management in the SAA environment is similar to software management and places additional loads on the configuration database. Because the upstream mainframe processor will be required to backup its associated departmental processors, critical downstream databases will have to be replicated upstream. These databases will have to be synchronized at intervals depending on the downstream transaction volume. This technique may also be used to provide database backup and recovery similar to offsite vaulting.

Performing the functions listed above represent a complex and pervasive set of tasks. To accomplish them in a cost effective manner will require extensive support in the areas of data acquisition, data management, data analysis, and reporting. The next section describes some of the characteristics of a system to accomplish these tasks.

6. THE SAA MANAGEMENT SUPPORT SYSTEM

Managing the "array" of processors and operating systems that make up the SAA environment requires a mechanism for consolidating data across processor vendors, models and system software environments. Providing facilities to store this data, make inquiries upon it, ensure its availability (backup and recovery facilities), etc. We will define this system as the SAA Management Support System.

The development of comprehensive specifications for an SAA MSS are beyond the scope of this paper however, the following paragraphs describe a few of the functional characteristics it will require.

The key to the effective and efficient utilization of the information collected within the SAA environment is correctly organizing the data. This organization should enable analyses to be carried out across multiple processors and allow comparisons to be drawn between dissimilar processors. In addition, the summarization methodology should allow for the consolidation of data across multiple dissimilar operating environments while allowing for the retention of data elements which are unique to each processor/operating system implementation.

Various I/S management functions will be controlled from various levels within the total organization. Few operational decisions will be made at the departmental level itself. However, many will be coordinated separately by each individual site. Some of these activities will be localized, but others will have to be reported to and coordinated with other sites throughout the organization. Thus a requirement is created for a comprehensive status reporting mechanism.

At a minimum, information related to the basic I/S management categories listed below will have to be collected.

- Configuration
- Workload
- Performance
- Availability
- Capacity

- o Reliability
- o Financial/Accounting

This information will have to be logged independently at the departmental, site, and establishment levels of the SAA hierarchy. However, the majority of its archival and processing will be carried out on the site and establishment mainframe processors. As a result, the system which manages and maintains this information will have to fulfill five key roles:

- o Data Management
- o Data Transmission
- o Data Consolidation
- o Data Storage
- o Data Presentation

Subsets of these functions will be performed at various locations both upstream and downstream. Figure 4 shows the relationship between these functions which are, themselves, described in detail below.

6.1. DATA COLLECTION

Data will be collected by monitor products which are unique to each operating system environment. The collectors will log data to files resident on the departmental processor for later transmission upstream through the network.

6.2. DOWNSTREAM DATA MANAGEMENT

The function of the Downstream Data Manager is to manage the collection and transmission of monitor log data. This includes switching the monitor log files and staging data for transmission to an upstream node. The Downstream Data Manager will ensure that no data is lost in the event of a communications outage or failure. In addition, it will reduce to a minimum the amount of manual intervention required at the departmental level.

6.3. DATA TRANSMISSION

Data logged at the departmental processor will be transmitted to the establishment processor via a process equivalent to a POS "pull". That is, the establishment processor will query the departmental processor for the available data which it will then transmit. The transmission will be accomplished by the Upstream and Downstream Transporters utilizing an LU6.2 session.

The Upstream Data Manager will cause the Upstream Transporter to initiate a request for data from a Downstream Transporter. The Downstream Transporter will send the data that the Downstream Data Manager has staged. Successful transmission will be verified by both transporters. The Upstream Transporter will notify the Upstream Data Manager of successful receipt of data and that node will be marked complete in the "pull" list. Upon verification of successful transmission, the Downstream Transporter will notify the Downstream Data Manager which will mark the dataset as sent. When all datasets have been successfully transmitted, the Downstream Data Manager will "clean-up" the staged datasets based on user-defined retention schedules.

6.4. UPSTREAM DATA MANAGEMENT

The Upstream Data Manager schedules and controls the data "pull" from the departmental nodes. It maintains a list of nodes and the associated times at which to "pull" them. It maintains a transmission log indicating which nodes were successfully "pulled" and which could not be contacted. The Upstream Data Manager can be parameterized to make multiple "pull" attempts if a node is not contacted.

6.5. DATA CONSOLIDATION

When the data "pull" is completed, the Upstream Data Manager will invoke the Upstream Consolidator to copy the "pulled" data into a single set of files for input to the Upstream Management Database. These files will be maintained as GDG's.

Using these facilities, an I/S management support database could be maintained at each site processor as well as at the establishment processor. A schematic representation of the relationships between the departmental processors and the site and establishment level management support databases is shown in Figure 5.

6.6. DATA PRESENTATION

The SAA Management Support System will have to present information in the appropriate format and at the appropriate scope and level of detail for each level of the SAA hierarchy. While the content of these information displays (tabular reports and graphics) will differ, their formats should be consistent. In general, these information displays will fall into one of six categories:

- o Management Alerts
- o Management Control Reports
- o Operational Exceptions
- o Operational Reports
- o Decision Support Analysis
- o Ad Hoc Inquiry

6.6.1. Management Alerts. are produced daily and monthly and identify variances from established management objectives.

6.6.2. Management Control Reports. summarize the activities of several functional units. They present status, planned-versus-actual and trend information.

6.6.3. Operational Exceptions. identify deviations from technical objectives to the operational support staff.

6.6.4. Operational Control Reports. provide status and performance information by technology group.

6.6.5. Decision Support Analysis. tools provide the facilities to manage I/S functions which cross operating unit boundaries or incorporate multiple I/S resources.

6.6.6. Ad Hoc Inquiries. are unscheduled and unstructured information requests. Because the inquiry process often involves several information extracts in which successive queries result from the real-time analysis of preceeding data, they place a premium on a flexible and responsive query facility.

7. SUMMARY

Twenty-four years ago, a major revolution occurred in the history of information systems, the announcement of the System/360. In light of the technological advances that have occurred since, in both hardware and software design, it is incredible that this basic architecture still supports the majority of commercial data processing worldwide.

As in most engineering endeavors, the compromises made in the 360 architecture reflected the economics of the

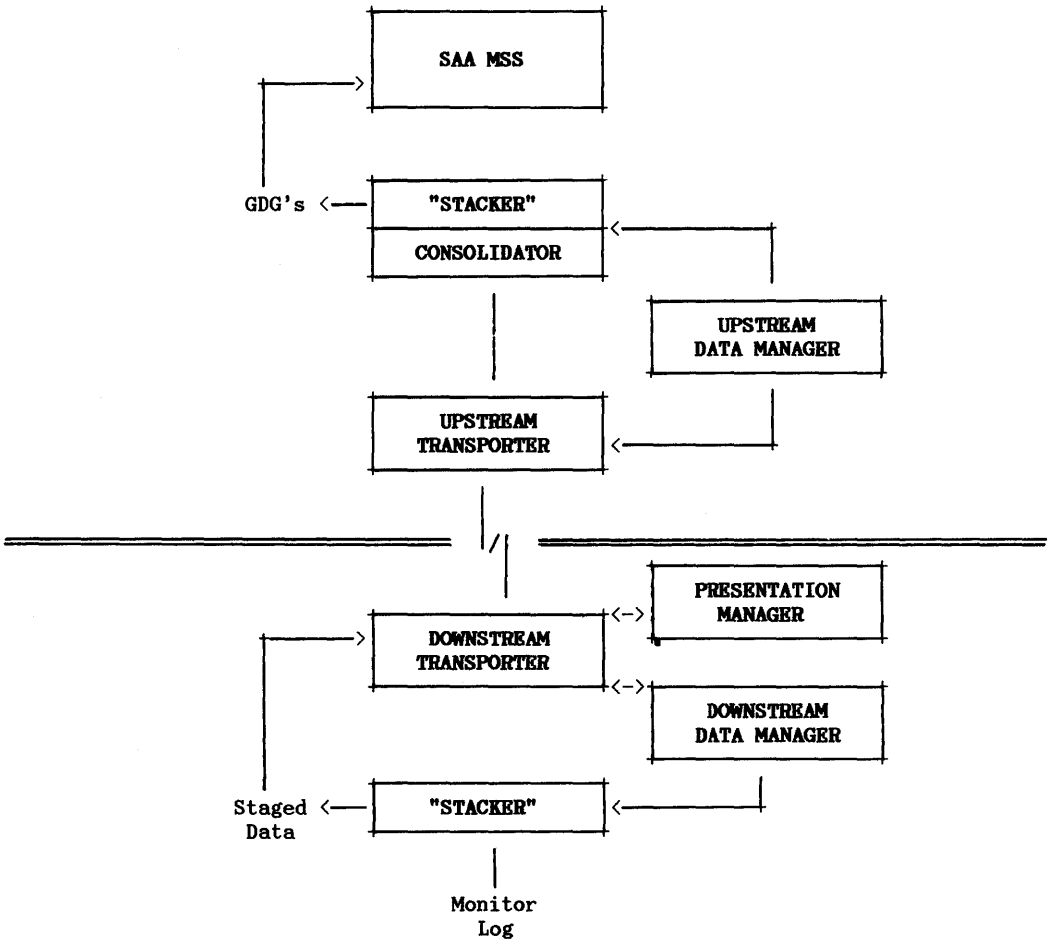


Figure 5. Functional Components of an SAA MSS

times. The advance of technology has shifted these price/performance boundaries. SAA is a reflection of these changing criteria. The implementation of SAA will, in some ways, be more painful than our predecessors' conversion to S/360 because we have so much more to convert. On the other hand, our experience, tools, and technology should help make it smoother and easier.

The decision to implement the SAA approach to I/S will not be an option for most enterprises. The direction of technological evolution coupled with the economic incentives will make this approach ever more competitively advantageous as time progresses. Therefore, I/S organizations need to provide now for managing these environments.

In the final analysis, most of us can only hope that SAA and the hardware and software designs it spawns will be as robust as the System/360 - 370 architecture it replaces. That way, it will probably be the only such conversion we ever have to make.

BIBLIOGRAPHY

1. Chester, Jeffrey A., "Departmental Computing: If it Were Only This Easy," *Infosystems*, April 1987.
2. Colony, George F., "New Ports of Call", *Datamation*, November 15, 1986.
3. Freedman, David H., "digital vs. IBM", *Infosystems*, May, 1987.
4. Henry, G. Glenn, "IBM Small System Architecture and Design - Past, Present, and Future", *IBM Systems Journal*, Vol 25, Nos. 3/4 1986.
5. Hopkins, Martin E., "A Perspective on the 801/Reduced Instruction Set Computer", *IBM Systems Journal*, Vol 26, No. 1 1987.
6. Kull, David, "SAA Master Plan or Grand Illusion", *Computer & Communications Decisions*, June 1987.
7. Lorin, Harold, "Systems Architecture in Transition - An Overview," *IBM Systems Journal*, Vol 25, Nos. 3/4 1986.
8. Matick, Richard E., "Impact of Memory Systems on Computer Architecture and System Organization", *IBM Systems Journal*, Vol 25, Nos. 3/4 1986.
9. Moad, Jeff, "Midrange Systems", *Datamation*, July 1, 1987.
10. Scherr, A. L.,"Structures for Networks of Systems", *IBM Systems Journal*, Vol. 26, No. 1, 1987.
11. Selinger, Patricia G., "Database Technology", *IBM Systems Journal*, Vol 26, No. 1 1987.
12. Strobl, Rudolf L., "The Flaws in SAA", *Computer & Communications Decisions*, June 1987.
13. Sundstrom, Robert J., *et al*, "SNA: Current Requirements and Direction", *IBM Systems Journal*, Vol 26, No. 1 1987.
14. Computerworld, *Focus on Departmental Computing*, Vol. 21 No. 18A, May 6, 1987.
15. The IBM 9370 Information System, Technology Brochure (G580-1004), International Business Machines Corporation, Armonk, N.Y., 1986.
16. IBM Corporation, *Systems Application Architecture An Overview*, GC-26-4341-0, May 26, 1987.

DASD MANAGEMENT: LET'S TALK MONEY

Greg H. Parker
Mountain Bell Telephone

ABSTRACT

Because disk related hardware represents a significant investment to the data center, the prevailing DASD management philosophy should be sensitive to business costs and corporate strategies. By successfully balancing the relationship of storage capacities, workload service, and configuration cost effectiveness; performance and capacity management activities can be performed within budget and still compliment company plans and objectives. Identifying the cost and expense associated with a given level of service is a key function in this type of DASD management program. Because costs play such an important role in the ability of the data center to provide the necessary service and data space capacity of it's major workloads, business cases are oftentimes required for resolving future needs. This paper addresses how costs and recurring expenses can be extracted and utilized to build a cost effective, yet service oriented DASD configuration.

1. INTRODUCTION

The term "DASD management" takes on a special meaning to different people. However, most would agree that in terms of cost and performance, the potential impacts of DASD makes managing this resource an extremely important function in the data center.

Many of todays data centers maintain comfortable levels of spare storage for reasons relative to performance, hardware failures, and anticipated application growth--while other centers operate at near capacity. Department budgets, company politics, and personal bias' are all controlling factors of how disk space is utilized.

With regard to performance, it is important to realize that the amount of data online contributes to device performance. We have to ask, "how much data can be placed on a device and maintain good performance?"[1] Furthermore, "how can the activity be spread over the configuration in order to provide priority workloads with the service required?" Some industry observers estimate that in a typical installation, about a third of IBM 3380 drive space is unused because of I/O performance considerations.[2]

Hardware failures are costly in terms of service and availability commitments. Experience seems to prove that a small percentage of empty spare volumes are essential for timely data recoveries in critical, high availability online environments. The choice of how many volumes depends on the configuration size, mix of device types, and nature of the workloads. Whereas, a minimum of four and a maximum of sixteen may seem reasonable, can the cost of this empty space be justified in terms of insurance?

Forecasting the storage requirements of future workloads is oftentimes frustrating. For example, a dependence on multiple application groups for anticipated growth projections, increases the uncertainty of space reserve requirements. Working within too strict of a DASD procurement policy, gives the appearance of being unsympathetic to user needs. However, a relaxed and over-obliging attitude lends itself to waste and inefficiency.

While the extreme in maintaining comfort zones of empty space for whatever reason can be a blatant misuse of company money, the result of exploiting every available byte of disk storage leads to the negligence of I/O performance. Thus, the DASD management process should be sensitive to costs while addressing the various functions of performance and capacity management. In this context, the overall goal is directed at building a cost effective, yet service oriented DASD configuration.

2. PRORATING DATA OCCUPANCY COSTS

The first objective is to identify in terms of capacity, the cost distribution of the DASD configuration. In other words, how much of the available DASD space is being utilized; what amount remains unused; and what is the cost of this condition? Additionally, we want to know what workloads are responsible for the resident data.

The approach used to break-out a workload's DASD costs begins with a baseline value consistent to both capacity and performance. This value is derived from the cost of all DASD channels, control units, cache, and disk devices in the system's hardware configuration. It should also be noted that the baseline value will be used for DASD management purposes--not for user billing. This value is referred throughout the remainder of this paper as the "baseline-cost."

Other recurring costs that can generally be found in department records are relative to vendor maintenance. These expenses can be tracked as a separate baseline value. Therefore, a workload can be held accountable for it's share of intermittent expenses in addition to the cost of the space consumed during peak DASD I/O activity. The term "baseline-expense" will be associated with this value.

Unlike a DASD billing product, where data residency time lengths are critical for effective charge-back, a peak period "snap shot" view of space utilization is used as input. High I/O activity time frames are used because of the response time effects of peaks to online systems.[3]

2.1. DATA COLLECTION

One or more local DASD management reports may be required to extract the necessary data. Volume information is needed in addition to measurements of data set utilization. Separate data bases can be easily created and later used for growth and trend reporting. The minimum variables required are:

VOLUME

- Unit Address
- Volser
- Byte Capacity
- Bytes Allocated
- Mount Attribute
- Date & Time

DATA SET

- Data Set Name
- Bytes Allocated
- Bytes Used
- Date & Time

2.2. COST-PER-BYTE

A cost-per-byte of DASD storage is derived from the earlier calculated baseline-cost and the byte capacity of the disk configuration. Each available byte of storage can then be expressed in terms of cost. Using the other baseline for expense, a byte level value can be generated that indicates the data centers on-going commitment for providing the existing space. To determine a cost-per-byte of DASD storage:

. Total the byte capacity of all DASD devices.

. Divide the baseline-cost by the total capacity.

The end result is fractional dollar amount for one byte of storage. Repeat the calculation using the baseline-expense value to determine an expense-per-byte of storage.

Since the scope of our analysis encompasses the entire DASD configuration, there is an assumption that resources are distributed in a priority fashion that provides each workload with it's required level of service. Therefore the assignment of resources and relationship between workloads is according to some master plan. This assumption precludes having to break down costs by device type even though there may be performance differences.

2.3. WORKLOAD IDENTIFICATION

In order to determine what workloads, or major applications are responsible for the allocated DASD space, it is necessary to find a common variable for grouping. Automated routines prove extremely helpful when processing large amounts of data.

Where data set naming standards are used, one method is to look at one or more levels of the name. Then a series of "if" statements can be used to assign an owner to the data set. For example, the following SAS statements can be used to extract the high level node of a data set name and assign it to a major workload:

```
DSNODE = SCAN(DSN,1,'.')  ;
IF DSNODE =  'WKL' OR
   DSNODE =  'WKLB'
         THEN WORKLOAD =  'APPLICATION A'  ;
```

The "if" statements can be expanded to include all existing high level nodes and all workloads. In shops where entire volumes are used exclusively by a single workload, volume records can substitute for data set records.

2.4. SUMMARIZATIONS

To establish the relationship between the major workloads and the configuration's available capacity, total cost, and total expense, it is necessary to perform high level summaries. Once complete, the results will contain the essence as to how cost effective the DASD configuration was at the time of data collection.

Using the volume level information, we can determine the total available byte capacity of the devices. Furthermore, it is possible to subdivide the summary total by device type and mount attribute. The cost-per-byte and expense-per-byte values can then be multiplied by the allocated and device capacity totals to determine the cost and recurring expense of each summarized category.

With the data set information, we can perform a similar summary routine at a workload level. The cost-per byte values can be applied to the total allocated bytes for the purpose of determining what part of the total investment is being used by the different workloads. Figures 1 through 6 are examples of how costs and recurring expenses can be implemented into management reports.

A cost utilization measurement can also be established by looking at the used and unused capacity relationship. The ratio is determined by dividing the unused, or still available space value by the used, or allocated value.

For example, given a DASD configuration consisting of 120 gigabytes of capacity with 80 gigabytes allocated, the cost utilization ratio (40 GB / 80 GB) would be 1/2, or fifty percent. This is another way of saying that one out of every three bytes of storage is left unused. From a pure efficiency viewpoint, a smaller percentage indicates better use of company money.

Managing DASD space in terms of dollars instead of strictly tracks or cylinders is a worthwhile endeaver. It not only captures one's attention but develops an appreciation and awareness for business costs. However, the cost of capacity is more tangible and easily understood by management than is the cost of performance. Establishing the price of good service is further compounded by factors such as inaccurate workload characterizations and the lack of specific service agreements.

3. PRIORITY WORKLOADS

A performance problem exists when response time commitments to resident workloads are not being met. Typically these commitments are in the form of service level agreements. Furthermore, the relative importance among applications is determined by the resource, service, and availability requirements agreed to by the data center's management.

An installations' priority workloads will typically be given preferential treatment in terms of faster I/O and minimum contention. These are the applications that tend to utilize solid state devices, cache, short strings, and partially allocated volumes. Therefore, when an important workload requires a specified level of DASD service, what becomes the most cost effective means for providing the expected response time?

Like any other system resource, increased demand for DASD can be satisfied three ways:[4]

. Buy it.

. Tune for it.

. Steal it from another workload.

However, each method has it's own associated cost. The reference of stealing resources from other workloads is a case of robbing the poor to give to the rich. Hence the cost is in terms of less resources and decreased service to the victim workloads.

Tuning is costly with regard to the time and energy of the people involved. This method attempts to utilize an existing configuration more efficiently and more effectively. Although time consuming, successful tuning efforts can provide significant hardware savings to the data center.

	TOTAL DASD INVENTORY VALUE	TOTAL DASD RECURRING EXPENSE	ALLOCATED DASD SPACE VALUE	ALLOCATED DASD SPACE EXPENSE	UNUSED DASD SPACE VALUE	UNUSED DASD SPACE EXPENSE	
	$2,150,402.00	$8,025.00	$1,686,756.00	$6,467.00	$463,646.00	$1,558.00	

FIGURE 1. COST SUMMARY

MAJOR WORKLOAD	CYLINDERS ALLOCATED	CYLINDERS USED	CYLINDERS IDLE	PERCENT OVER-ALLOCATED	COST OF ALLOCATED RESOURCES	RECURRING EXPENSE	PERCENT ALLOCATION GROWTH
PAYROLL	24683.1	19613.8	5079.3	21	$550,010.00	$2,183.88	-15.30
TOLL	51538.0	39861.7	11676.3	23	$746,496.00	$2,964.05	32.23
SYSTEM	37972.6	29709.6	8263.0	22	$390,250.00	$1,319.07	3.24
	=====	=====	=====		========	======	
	114193.7	891855.1	25008.6		$1,686,756.00	$6,467.00	

FIGURE 2. MAJOR WORKLOAD SUMMARY

DASD DEVICE TYPE	TOTAL VOLUMES	PRIVATE MOUNTS	STORAGE MOUNTS	PUBLIC MOUNTS	PERCENT USED	PERCENT FREE	INCREASE IN TOTAL VOLUMES
3350	32	29	3	0	85	15	2
3380	180	164	10	6	79	21	8
	===	===	==	=			==
	212	193	13	6			10

FIGURE 3. VOLUME SUMMARY

DATA SET NODE	MAJOR WORKLOAD	COST OF ALLOCATED RESOURCES	RECURRING EXPENSE	PERCENT ALLOCATION GROWTH	BYTES ALLOCATED	BYTES IDLE	PERCENT OVER-ALLOCATED
PAY	PAYROLL	$173,029.51	$687.03	-28.48	8507176964	441052040	5.2
PAYDB	PAYROLL	$113,248.59	$449.67	19.03	5567985280	987880608	17.7
TOLLDB	TOLL	$229,186.92	$910.01	5.77	11268214772	4559802982	40.5
...	...	...	...	...	...	...	...

FIGURE 4. ALLOCATION COSTS GREATER THAN 100,000 DOLLARS

DATA SET NODE	MAJOR WORKLOAD	COST OF ALLOCATED RESOURCES	RECURRING EXPENSE	PERCENT ALLOCATION GROWTH	BYTES ALLOCATED	BYTES IDLE	PERCENT OVER-ALLOCATED
BKUP	SYSTEM	$98,911.43	$458.03	80.01	7987964	7652438	95.8
SYS87147	SYSTEM	$6,089.55	$91.17	0.00	244550	244550	100.0
TSTDB	TOLL	$32,333.02	$161.41	16.55	633762	629804	99.4
...	...	...	...	...	...	...	...

FIGURE 5. SPACE OVER-ALLOCATIONS GREATER THAN 95 PERCENT

DATA SET NODE	MAJOR WORKLOAD	COST OF ALLOCATED RESOURCES	RECURRING EXPENSE	PERCENT ALLOCATION GROWTH	BYTES ALLOCATED	BYTES IDLE	PERCENT OVER-ALLOCATED
ASB1	TOLL	$391.08	$1.55	0.00	19227780	5602168	29.1
ASBDB	TOLL	$118.77	$0.64	0.00	644550	113567	17.6
CRB	PAYROLL	$1,098.89	$4.23	0.96	40002212	887273	2.2
...	...	...	...	...	...	...	...

FIGURE 6. CUMULATIVE DATA SET NODE SUMMARY

once tuning and stealing for additional DASD resources have been fully exploited, buying more is the next alternative. Depending on the nature of the situation, procurement alternatives may include both hardware and software. For example, the reduction of physical DASD I/O provided by expanded storage, cache, and virtual fetch products may prove more advantageous than new hardware. Traditional hardware procurement solutions can overlook the benefits available with other viable alternatives.

3.1. BUSINESS CASE

Because of the expense associated with cache and semiconductor disks for example, justifying the cost may not be easy because it means assigning a dollar value to a soft benefit--better response times.[2] Most likely, before management will accept and approve the procurement of "performance boosters," a specific business case will need to be developed.

The document should include information on the configurations current performance and cost utilization. Furthermore, a specific tuning agenda containing the projected results will probably be required. Additionally and most important, is the presence of alternative solutions to a performance problem--each in relationship to workload service and cost to the company.

4. SUMMARY

The data center's DASD management philosophy should be concerned with storage capacities, workload service, and configuration cost effectiveness. Furthermore, performance and capacity management efforts work best if all possible solutions are investigated. As each alternative is reviewed, the preferential choice will be the one that meets the service level requirements and provide the most benefit to current and future business plans--with the least cost.

A DASD configuration's cost and recurring expense are important baseline, or index values for managing disk space. Whereas dollar values command greater attention than traditional measurements of tracks and cylinders, it also places management reports at a level more appropriate for business planning. Additionally, a sense of cost effectiveness can be developed based on how well existing resources meet organizational objectives.

The overall goal of DASD management should be to provide a cost effective, yet service oriented disk configuration that compliments company business plans and philosophies. Because the data center's disk investment deserves the same attention and financial care as other major corporate assets, an awareness of costs and expenses is essential.

REFERENCES

1. Friesenborg, S. E. / Wicks, R.J., DASD Expectations: The 3380, 3880-23, and MVS/XA, IBM Publication GG22-9363-02 (August 1985)

2. Kull, D., Busting the I/O Bottleneck, Computer & Communications Decisions (May 1987)

3. Kinsbergen, J., The State of Current-Time Management: Tune for Service Not Throughput, Boolean World (Winter 1986)

4. Friesenborg, S.E. / Hall, G.L., MVS Performance Management, IBM Publication GG22-9351-00 (January 1985)

WORKLOAD CHARACTERIZATION AND FORECASTING FOR A LARGE COMMERCIAL ENVIRONMENT

George W. (Bill) Miller
American Airlines

ABSTRACT

This paper suggests an approach for characterizing the activity of a large data processing environment into discrete workloads which are consistent, measurable, and recognizable. Some practical techniques are suggested for predicting the growth of these workloads using a combination of historical trending and the identification of major future influences. The discussion is conceptual in nature and no detailed technical expertise is required other than a general understanding of the nature of a computer center.

INTRODUCTION

It can be truthfully said that all forecasts are wrong. The challenge is to reduce the margin of error down to an acceptable level. The approach used will depend on the size and complexity of the installation, the nature of the business, and the historical information available. In simplest terms, workload characterization is a way of describing the DP activity occurring in an installation. Forecasting involves determining what has happened in the past and predicting how the future is expected to differ from that.

WORKLOAD CHARACTERIZATION

OBJECTIVES

The computer workload of a large data center is most likely to be both varied and complex. In all probability there are both batch and interactive environments coexisting in some fashion. There will be hundreds or thousands of different jobs and transactions running at all hours of the day and night, doing very different things, and having very different resource requirements. In order to be able to describe all of this we must develop some means of grouping all of this activity into a manageable number of pieces.

The object of any workload characterization technique is to group elements of the workload that collectively represent some consistent resource characteristics, are distinctly visible to the observer, and which ideally have some intuitive meaning to the observer.

Consistent. The characterization must display consistent resource characteristics from one observation to another. One would expect to talk in terms of resources for a given volume of activity. The individual elements which have been grouped together may or may not have similar characteristics; the group as a whole must have.

Measureable. It must be possible to discreetly identify and measure the elements which make up the characterization. They must be distinguishable from other elements of the workload with no overlap or missing pieces. This implies that appropriate accounting information is in place and routinely available and that the necessary software is in place to process and retain this information. Many authors have detailed the possible sources of accounting information and explained in detail the techniques for processing this information. A lengthly discussion of SMF data, RMF data, IMS log files, etc., will not be repeated. Suffice it to say, however, that the existence of dependable historical performance data is the stuff of life here.

Recognizable. If the pieces into which the workload has been grouped are not intuitively recognizable to the rank and file within the installation, then discussion of present and future activity in terms of these pieces does not necessarily accomplish good communication. Management is better able to discuss and understand the need to accommodate growth of a particular workload if it represents something whose function and purpose is understood.

UNITS

For purposes of discussion it seems useful to identify three kinds of units:

Computer Resource Units (CRUs) are the basic units representing computer hardware activity. Examples would be CPU time, various I/O counts, tape mounts, DASD storage, etc. Service Units (SUs) are weighted composites of several CRUs and are used to describe the overall hardware resource associated with some activity.

Computer Workload Units (CWUs) are the basic workload elements within the computing environment. Examples are batch jobs and interactive transactions. The resource characteristics of CWUs are described in terms of CRUs, i.e., we describe the activity of a batch job in terms of the volume of CPU time and I/O counts.

Business Work Units (BWUs) are the business related activities that are the purpose of all this computer activity. Examples are bank deposits, claims received, units manufactured, or passengers boarded. Until we look at the work done in terms of BWUs, we have difficulty in distinguishing the computer installation of a bank from that of an insurance company or a manufacturing company or an airline. These BWUs will require a number of jobs and transactions (CWUs) for their completion and will ultimately be equated to typical requirements for resources (CRUs). Note that what is being called a Business Work Unit here is the same thing that other authors have referred to as a Natural Forecast Unit (NFU).

GROUP WORKLOAD ELEMENTS

In most cases workload characterization is simply a technique for grouping Computer Workload Units. Each of these techniques has both advantages and disadvantages.

Clustering. Techniques can be used to identify elements of the workload with like or similar resource characteristics. One example might be that set of batch jobs which typically account for less than 10 seconds of CPU time, no tape mounts, and produce less than 50 pages of printed output. Another might be an IMS transaction with less than 2 seconds of program execution time and less than 25 DBD calls, or whatever. This results in a characterization with very consistent CRUs. The problem with using clustering to characterize the workload for purposes of workload forecasting is that the resulting groups may be intuitively meaningful to no one except the performance

analyst. Clustering is a technique particularly useful for benchmarking and modeling.

Major Application. By "application" is meant a functionally related set of work elements such as the jobs and transactions making up a payroll system. Although the elements being grouped together are likely to vary considerably in their individual resource characteristics, the typical CRUs for the total application may be quite consistent. If so, then this has the advantage that the application represents a piece of work which has a recognizable identity to individuals within the organization. It may be possible to utilize survey or interview techniques to discuss the expected growth of an application. It may also be that the application is utilized by a number of organizations in several locations making such discussion more challenging. In a large installation there may be so many hundreds of major applications that the process becomes somewhat difficult to manage. Some additional higher level summarization may be necessary. This also implies that some accounting conventions or naming standards exist which will allow this activity to be properly identified from SMF or other accounting or log files.

Organization or Location. If considering the workload by application describes what the installation is doing, considering the workload by organization or location describes for whom it is being done. Again, the accounting information must exist to determine the who and where. If a chargeback system is already in place, this type of accounting should be straightforward. As before, there must be some consistency of overall volume and resource units, and there are apt to be problems with exceptions. There is also the problem of the work which does not fall neatly into a particular slot, for example development work, or system support. However, this characterization might be particularly useful in considering growth resulting from organizational or geographical expansion.

Major Subsystem. Not uncommonly, the workload is characterized by major subsystems such as batch, TSO, IMS, etc. This is perhaps the simplest characterization in terms of data collection. The organization of SMF and RFM data lends itself to a simple summary of subsystem activity without regard for who or where. This is also probably the most common way of characterizing the installation workload. These are the major pieces of work in an installation, and in a multiple CPU environment they will be placed so as to minimize conflict. Often the large interactive environments are kept to a single processer, and it is the projected growth of one of these to above a threshold level that signals the required upgrade of a mainframe.

Installation. Equally simple is the compilation of total resource for the installation as a whole. This yields the grossest characterization of them all, the total shop. This is in fact the bottom line. Developing history and an overall growth trend of total installation gives one a target or check for the process of adding up the pieces. If the sum of the pieces does not fall near the trend line, it would be interesting to know why not.

Business Work Units. Ultimately one would prefer to use Business Work Units to characterize the workload. This will serve to tie the data processing forecast to the business forecast and makes the DP forecast much easier for corporate management to understand. The problem with BWUs is that there are often large segments of the installation workload that do not tie neatly to fluctuations in these BWUs. The alternatives are to show these activities separately as "overhead" or "installation support" or to prorate them somehow over those activities that do relate to BWUs. If the resources associated with the activities in question are considerable, neither of these alternatives are apt to be entirely satisfactory.

USERS/CUSTOMERS

A typical large computer installation is putting out an impressive amount of DP resource to somebody. It would be useful to know who that somebody is. While it is relatively easy to know what the impact of the major workloads is, it is not as commonly known just who is responsible for all that activity.

If you are in the business of selling computer service to outside customers then you will have developed some means for determining the cost of the service provided to each customer and the means for billing them for it. If not, then your career as a service bureau is likely to be somewhat short lived. Even if the DP center provides service only to itself, or to the larger organization of which it is a part, there is a requirement to define and quantify the cost of services to each user or customer. Any DP organization worth its salt will have some kind of a process for doing just this, be it called chargeback, cost allocation, transfer pricing, or whatever. But what if such a system is not present? How might reporting of resource by customer be developed for management control?

*BRIDGES

Most installations have developed the means for doing some kind of workload summarization for management reporting purposes. What is required is a relationship or "bridge" between the activity described in the basic accounting log record (SMF or IMS log, etc.) and the identity of the group (the particular workloads, applications, organizations, locations, customers, etc.) to which this activity is to be related.

Stated very simply, the process consists of three steps. First develop the means for obtaining the basic accounting information which describes detailed computer activity. Second, describe the "buckets" into which you want this detail accumulated. Third, build a bridge of relationships which will tell you into which bucket to pitch this detail.

If these source records contain good accounting codes which have been well thought out, enforced, and maintained, then you should be home free. A chargeback or cost allocation system essentially solves the problem. If you are one of the multitudes of shops whose accounting codes are a series of poorly maintained good intentions, then you have a challenge. Even so, the lack of a convenient mechanism for doing something does not remove the need or benefit of doing it. Nor will it necessarily get you off the hook.

If a proper and convenient bridge does not exist, then one must develop what the military calls a field expedient. The use of battlefield terminology is not completely inappropriate here. To group activity by application, first go talk to your scheduling people. They have to know what jobs go together. The data base administration people can do the same for IMS, CICS, etc. It may be necessary to build tables of job names or transaction names to application systems. It may also be that someone has already worked out such a relationship and even maintains it. It is likely that several relationships will have to be tied together to eventually get from the accounting source to the proper bucket.

To relate activity to organization or geographic

location, again consult the good people in operations. They have to send all the computer paper to somebody. The people responsible for the interactive systems should know who and where the users are. The source records will likely contain some sort of user ID required for logon or for data security. Either the system support or data security people can no doubt relate these user IDs to a name and or an employee number, etc. From here the personnel or payroll people can point to organization and/or location. Try the people who maintain the organization phone book for a relationship between names and places. the idea is that relationships will exist or can be created which can be pieced together to build a bridge from the source data to the desired grouping. Once the relationships have been established the intermediate steps can be eliminated.

Resourcefulness and creativity are an asset here. It is also very helpful to ignore the advice of knowledgeable and well-meaning individuals who counsel that this sort of thing will not work, and continue to snoop about.

REALITY

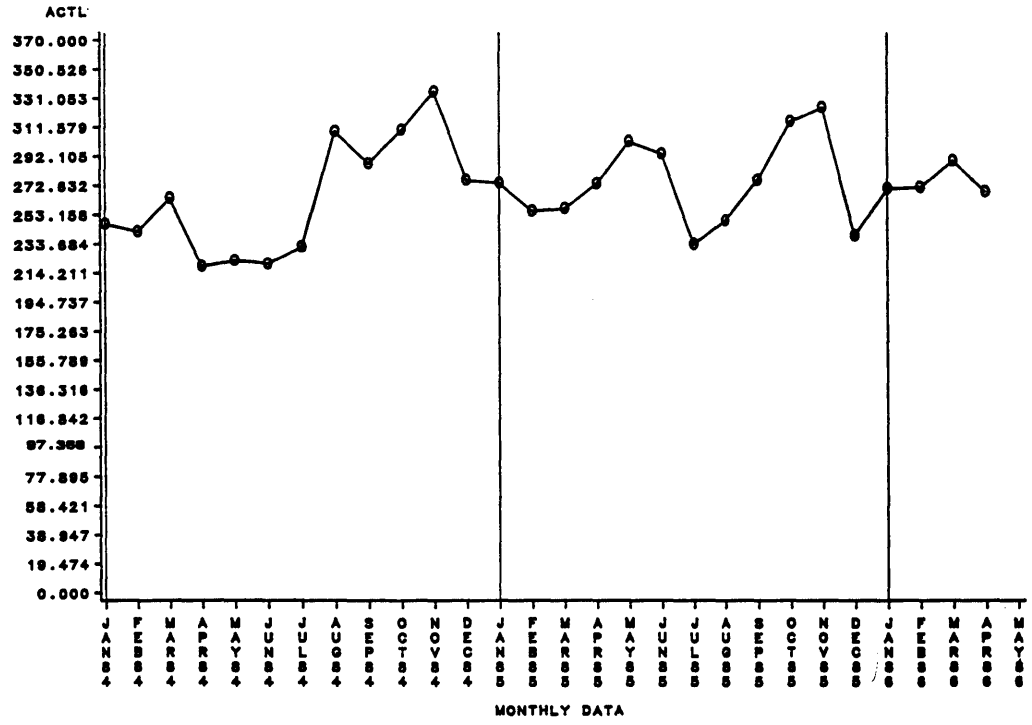
Having set out a nice neat list of schemes for characterizing the computer workload, it is only fair to admit that none of them is really completely adequate. The complexities of our environments prevent any one approach from fitting all the odd cases without a good bit of stretching and hand-waving. They serve different purposes and answer different questions. To develop a forecast whose probable margin of error is within an acceptable comfort range, one must characterize and forecast the workload in several different ways. What emerges is a final result which hopefully is both reasonably accurate and intuitively sensible.

BUILDING THE FORECAST

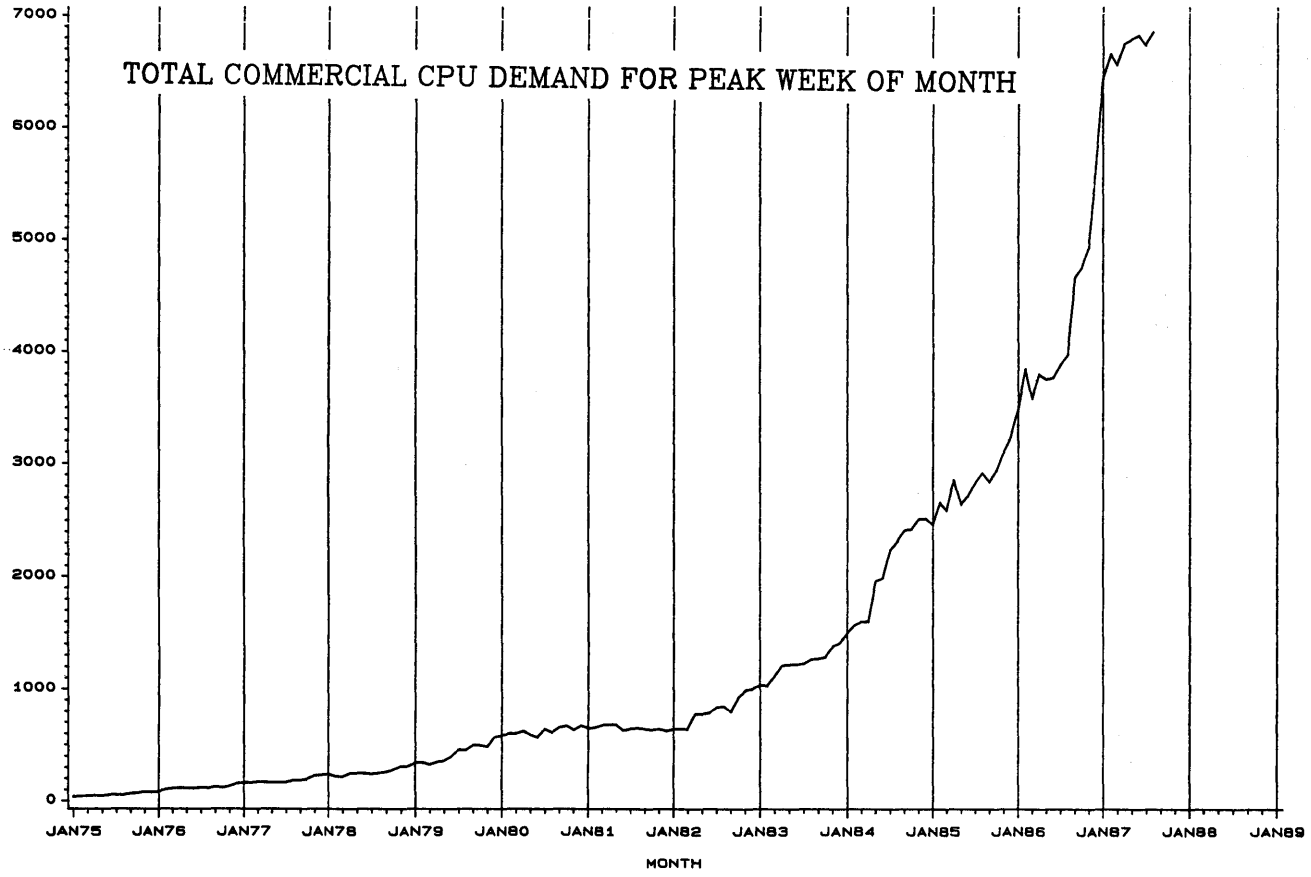
Let us now investigate the black art of computer forecasting. The simplest but not necessarily most cost effective way of forecasting is to call in your friendly neighborhood hardware vendor and ask for a recommendation. This will be almost guaranteed to solve your capacity problems. Just in case this is not what you want to do, we will explore some other options.

MACRO FORECASTING - The Big Picture Approach

History. The first step would seem to be to get an overall understanding of the historical growth patterns for the installation. It would be helpful to identify the total size of the installation in terms of MIPS or equivalent CPU numbers of some kind for as far back as feasible. The previous two years is minimal, the previous five is much better. The appropriate unit is anything that indicates the total shop across various mainframes. If continuous data is not available then interpolate between known points. Use graphs and charts profusely. Try to get an understanding of what the activity of the installation looks like. One may look at individual workloads (figure 1), or at the installation as a whole (figure 2). Actually, it is useful to do both.

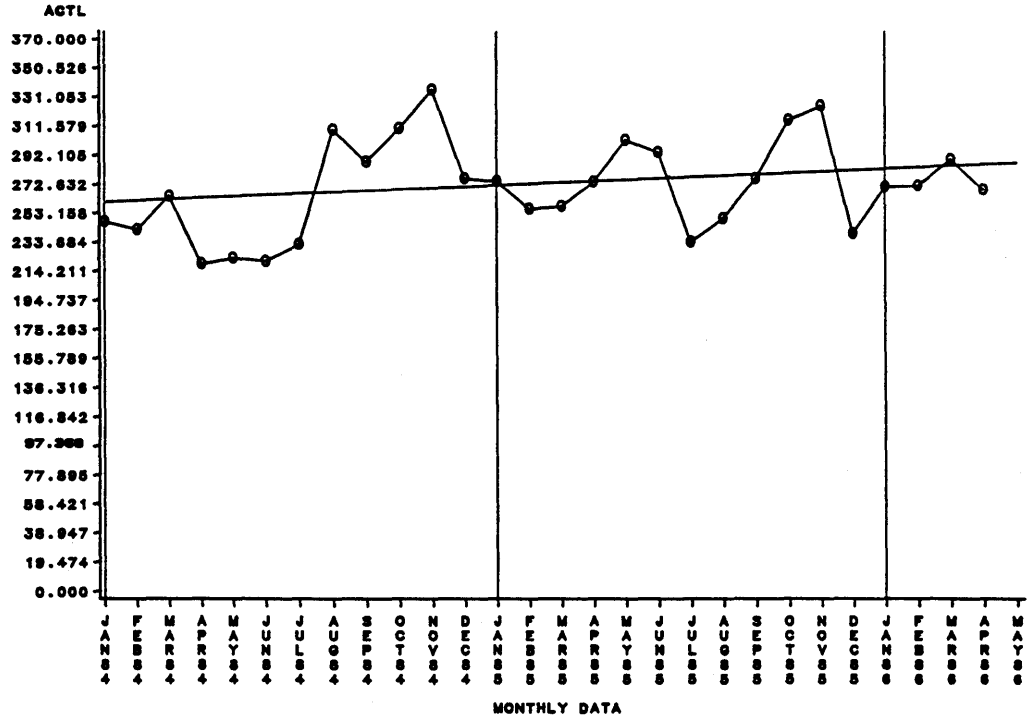


GROWTH OF WORKLOAD A
figure 1.



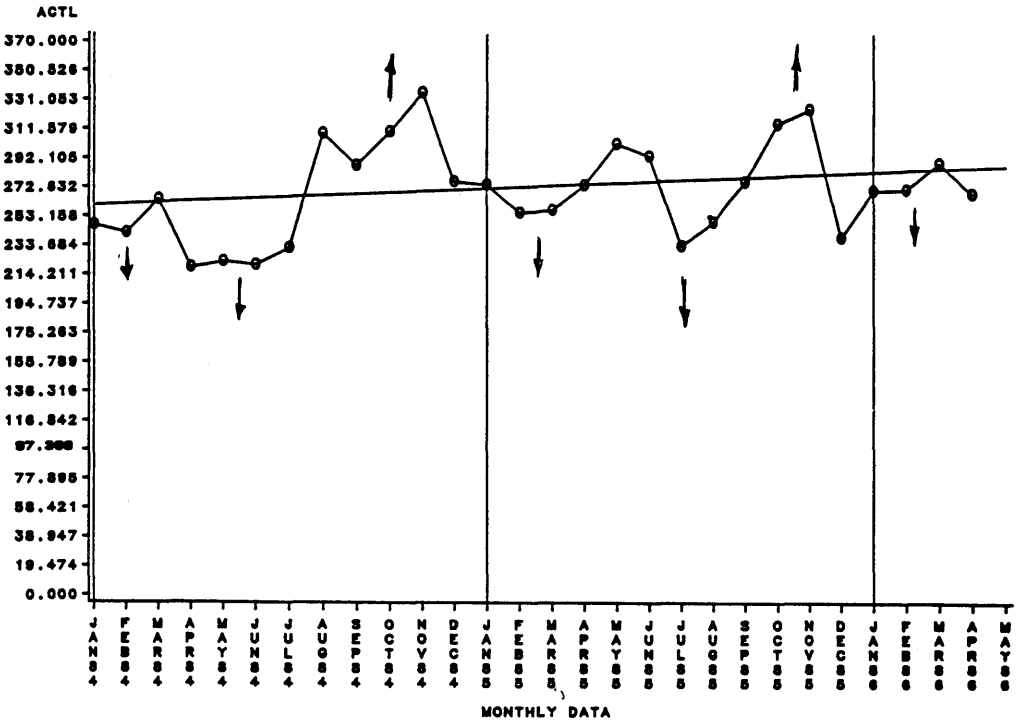
TOTAL INSTALLATION GROWTH
figure 2.

Trends. Develop some simple trends based on this historical information, then graph it and look at it. The typical reaction is to be somewhat frightened by the prospect of continuing to grow at the present rate. Remind yourself that the average large installation is growing at an annual rate of somewhere around 40%. Try to relax. So as not to frighten the reader, we have elected to look at Workload A (figure 3) rather than the total installation. Besides, Workload A is quite cyclic and makes a good example.



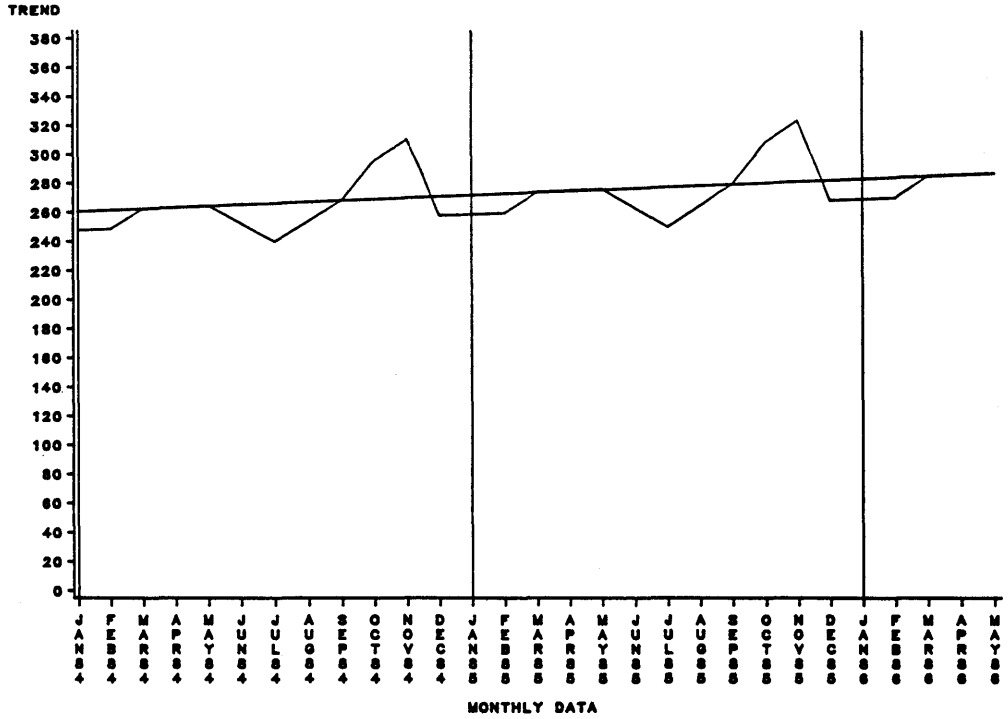
TREND OF WORKLOAD A
figure 3.

Variations. Over long periods of time, workloads tend to rise. From month to month, they tend to rise and fall, and viewed from week to week or day to day, they tend to go all over the place. Note periods of rise and decline and make a sincere attempt to be able to explain why they occurred in terms of hardware modification, major changes in the workload, and cyclic phenomenon (figure 4). Consider the relationship to business or budget cycles, seasonality influences, employee holiday periods, etc. If possible develop the same information for the total volume of batch and interactive workloads. The idea here is to develop a relatively simple model which will allow us to restate the past in somewhat abstract terms.



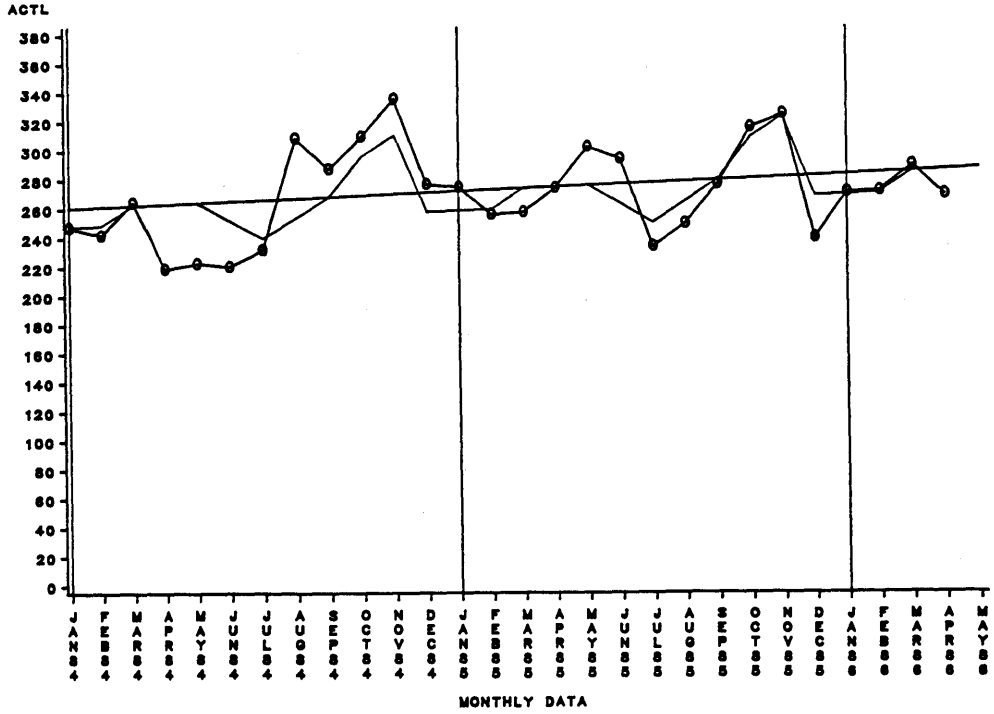
CYCLIC VARIATIONS IN WORKLOAD A

Adjusted Trend. Determine which of the variations are likely to continue to occur in the future, and apply them to a projection of the trend line (figure 5). If nothing is going to change and the future is expected to look like the past, then this adjusted trend line might serve as a forecast. It can be fairly noted that the future generally brings a few changes.



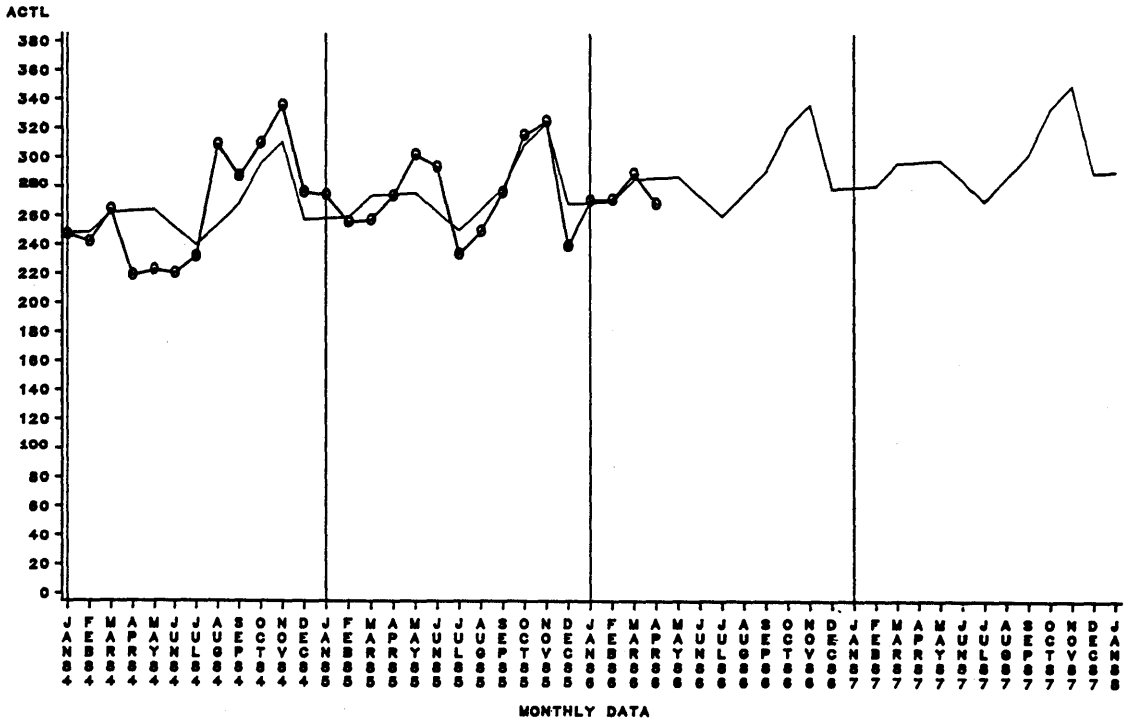
ADJUSTED TREND OF WORKLOAD A
figure 5.

Restate the past. If we have done this correctly, we will have constructed a graphic model which has approximately restated the past (figure 6). If the model will not restate the past, it will surely not predict the future. Furthermore, this model, while mathematically very simple, is intuitively quite satisfying.



SIMPLE MODEL OF WORKLOAD A
figure 6.

Major Changes. Review the business plan and interview the necessary people to determine the major changes in the environment which will occur over the next two to five years. You have determined what the past looks like. Now you are trying to get a general idea of the major ways in which the future will differ from the past. Estimate as best as possible the effect that these changes will have on the trend line. Again, graph and review, adjust and repeat. As this adjusted trend line is extended out into the future, it is a forecast with which we may have some relative degree of comfort (figure 7).



FORECAST OF WORKLOAD A
figure 7.

Projected Total Growth. The idea of this process is to gain a clear understanding of the nature of the growth of the installation and a general idea of where it is going. If this were to be used as the forecast, it shouldn't really be that far off. At least it should be close enough to define the ballpark. Further efforts to characterize and forecast the workload should not vary wildly from this ballpark without raising serious questions.

This might actually provide us with a forecast that we could live with if the resulting accuracy proved decent. In a large stable environment is can actually result in a forecast which is quite good. the problem with this is that it is simply not defensible. It has been developed from a somewhat lofty vantage point and is not based on detailed analysis of the many individual elements which make up the workload. It is based only on a knowledge of the past and the insight of a few well-informed people. Not bad, perhaps, but risky.

MICRO FORECASTING - Adding Up the Pieces

Knowledge of future activity can be obtained in a number of ways. Basically the challenge is to ask the right people the right questions at the right times. If we manage to get the right answers, forecasting is a simple matter of adding up the pieces to determine future requirements. This turns out to be simpler in concept than in execution.

Asking The Right People. Really, what we are trying to do is to identify those things that change. If nothing changes, then next year will look pretty familiar. The right people to ask are those who are in the position to discuss changes in the workloads. Obviously these people and their positions in the organizational structure will vary from place to place, but we can consider some examples.

The users/customers of the DP service can describe their plans for changes in the way they do their business. They may foresee increases or declines in certain areas or changes in the way they plan to utilize the services of the DP center in the future.

Most organizations have a group of marketing people who are aggressively trying to sell the services of the business, or of the data center directly. These people are paid to sell, not to minimize DP headaches, and at least occasionally can be counted on to make promises first and check on capabilities later. They often have several possibilities going concurrently and frankly are not always in the position to predict just how things will work out. It is also true that these people are generally difficult to gain access to and may be less than enthusiastic about discussing the possible outcome of some delicate negotiations. Nevertheless, these activities can have a profound effect on the DP workload and their information is of great value.

In many organizations the applications development people reside at the neck of the funnel between the customers and the operation of the data center. Applications development has the challenge of translating the requirements of the customer into an orderly subset of the workload which is compatible with the operation of the data center. For this reason, they can often provide excellent information on the characteristics of new applications and changes to existing applications.

Long range plans are probably best defined by data center or corporate management of director level or above. These are the big picture people who influence the emphasis of one particular environment or another, and who are privy to corporate strategy. Their time is valuable and so is the information and insight they can give. It is very important that these people understand the importance of your requirement for this information. The reason we need this information is to allow the forecasting and planning to be done which will provide them and their people with the computer resources required to do their jobs. The understanding and cooperation of these people is vital. Remember that the rest of the people we need to talk to report to them.

It also needs to be recalled that it sometimes pays to have friends in low places too. As time goes by, it will become apparent that there are certain individuals within the organization who are well placed to identify what is going on and what is likely to happen. These are apt to be very skilled technical people who may or may not have any managerial responsibility, but whose insight and perspective is invaluable. These are the people who can shed light on reality in a very helpful way. Cultivate their friendship. Buy them coffee. They can save lives.

Asking the right questions. What questions we ask depends on who we are asking. Asking customers what they are going to do assumes that they are in the position to know. It is possible that the activity of some customers will be driven by events beyond their control or current knowledge. It is not reasonable to ask a customer of DP services how much CPU time, or how many DASD EXCPs, or how many tape mounts are going to be required to support the customer's workload. It is reasonable to ask how much work is expected to occur in terms of Computer Workload Units or Business Workload Units, but translation of these requirement into equivalent resources is and should be the job of the performance analyst.

Personal interviews are perhaps the best means of obtaining information from marketing people and from upper management. The absence of an obvious written document should not indicate the lack of a specific agenda. Gather as much information as possible about new applications, expected changes to existing applications, expansions in business, new locations, increases in numbers of people having access to systems, etc. Remember that this is the ultimate opportunity to sell the individual on the importance of providing the best information possible. The better the information, the higher the probability of a smooth transition into the future.

The lack of hard information may cause an individual to hesitate to respond. The inability to predict the future with absolute certainty does not relieve the necessity of producing a forecast. If a considered best guess is all that is available, then that is what must be obtained. A best guess is almost certainly of greater value than no information at all. The alternative is terminal paralysis.

A periodic survey document may be used to obtain information from the users or customers of the DP services. Survey documents can be used to ask users about changes in their expected demand. Asking people what will be happening in the future assumes that they know what is happening now. It is not reasonable to ask what future activity will look like without providing information regarding the current level of activity and how that activity has varied over some period of recent history. The more information that is made available to the respondent, the better the information that can be provided by the respondent. If the base is properly established, response can be stated simply as a percent change. A good example of such a document can be found in reference [6]. it will not uncommonly be useful to follow up the survey with

an interview if the information warrants further investigation.

Resource Impact Forms can be developed and required as part of the development cycle of new project by the applications development people. Since the development people have the unique position between the customer and DP operations, they are often able to describe the characteristics of future changes to the workloads. If a completion of a comprehensive set of documents is made a part of the development cycle, much valuable information can be obtained. An example of such a document is referenced as attachment A.

The Right Time. The best time for obtaining forecast information is any time it is available. Resource impact forms established as part of the development cycle will likely be filtering in on a constant basis. Marketing and management interviews and surveys of the customer base might best precede the forecast cycle. It is necessary to constantly keep an ear to the ground to listen for distant rumblings. Keep the faith. There will come a time when the organization's DP community will understand the importance of this information in assuring that their own DP needs will be met, and will seek us out to see if we want some information that they think we may find useful. At that point we will know that we are getting somewhere.

Locating All The Pieces. At this point we have asked all the right people all the right questions, and have left no stones unturned. Now what do we have? Determining the total workload requirements by adding up all the pieces assumes that it is going to be possible to identify all the pieces. While we may be able to obtain a fair estimate of the expected workload activity for as much as 6 to 12 months into the future, between 12 months and 2 years things get pretty thin, and beyond 2 years falls into the realm of pure speculation. It is entirely possible that after taking all the surveys and adding up the pieces, the result indicates a declining total. Such a result should be viewed with a certain amount of suspicion. Again, it is most important that the estimates be based on knowledge of the existing workload. Careful consideration must be given to the growth of program development, system support, and installation management requirements. Some means must be devised for migrating into that hazy area of 12 months or more into the future. Those organizations that rely heavily on this approach tend to do a good bit of quiet gap filling.

A COMBINED APPROACH

The exclusive use of trending, and the attempt to identify and add up all the pieces form the extreme ends of a continuum. Obviously, what works is somewhere in between. It would seem sensible to define some middle ground where the strengths of each approach are combined. Actual experience has indicated that overall validity of the forecast has improved as we have shifted more towards a macro forecast. While we conscientiously gather detail on new activity from a variety of sources, we tend to apply that information only as it will pull the workload out of the "normal" range. Let us review how one might actually go about doing all this.

RESTATE THE PAST

Characterize The Workload. Decide on the workload characterization that best meets the needs and realities of your installation. This characterization should be consistent, measurable and recognizable, and some history should be available in these terms.

Create A Baseline. In any case it is advisable to first create our macro forecast. When considering the period from two years out, this would seem to be the best available. Even for the short term, the vagueries of the micro approach make the creation of this baseline necessary to define our ballpark.

Normal Growth. The trended value includes normal growth. This means that just as new applications were developed and implemented, and existing systems expanded and grew, the trend reflects the expectation that new applications will continue to be developed and existing systems will continue to expand at this same rate. This is "normal" growth. This reassures us that it is not necessary to identify every little piece of future work.

Predictable Cycles. Develop adjustments to the trend line based on those phenomenon which tend to push demand up or down at the same time each year. We are now able to restate the events of the past with a reasonable degree of accuracy. If no big changes to the workload or environment occur, this will also project the future.

IDENTIFY FUTURE CHANGES

Gather Information. Utilize the techniques of the micro approach to forecasting to gather as much information as possible about the ways in which the future will differ from the past. A variety of sources can be employed to gather revelant information. Survey documents can be used to ask customers about changes in their expected demand. Resource Impact Forms can be developed and required as part of the documentation of new projects. Key people both in high and low places should be periodically interviewed. Good 'ol boy networks should be cultivated. Experience is essential and an element of espionage may be helpful. Remember that people must be aware of their current level of activity before they can be expected to be able to describe their estimate of future activity. The means must be developed to relate these changes to workload and resource demand.

Publish Assumptions. Compile the significant results of the information gathering process into an assumptions document that basically indicates what changes are going to be considered in the next iteration of the forecast. Publish this document to a wide audience and wait for people to tell you why it is incorrect. Thank them.

Significant Changes. We must be careful to survey the right people with the right questions, but now we can recall with some comfort Pareto's 80/20 rule and concern ourselves most with those elements or combination of elements which will take us outside the "normal" range. Recall that "normal" growth has been accounted for in our trend development process. In fact, we may be tempted to deviate from our baseline forecast only when we are able to identify specific reasons for doing so. When we identify activity that will cause workload demand to fall outside trended values, we will indicate a step in the trend line.

What this amounts to is an admission that we are willing to go to a great deal of trouble to gather a volume of information that will largely be ignored. By doing this however, we have significantly reduced the possibility that a major impact on the accuracy of our forecast has slipped by undetected.

The Result. Now we have a forecast which is firmly based on reality. It is historically sound. It has identified and considered the best available information regarding changes to the environment and

the corporate business plan. It is an accurate as possible, and it is defensible. The ability to defend or explain a forecast in terms that are intuitively sound is obviously of great benefit. Ask anyone who had to renegotiate an ever-expanding DP budget with cost conscious management. Now we take a big deep breath and publish our forecast to the world, or at least to the same audience as we asked to review our assumptions. Don't forget computer operations and the people responsible for doing the hardware plan. They have a vested interest.

The final forecast document might look something like figure 8.

FORECASTED COMMERCIAL DEMAND
PREPARED BY RESOURCE MEASUREMENT & FORECASTING

INTERACTIVE SYSTEMS

MONTH	TSO PEAK WEEK	TSO PEAK HOUR	TSO AVRG WEEK	TSO MAX USERS	CPU PER KTRAN	TSOE APL PEAK WEEK	TSOE APL AVRG WEEK	CPU PER KTRAN	TSOE DB2 PEAK WEEK	TSOE DB2 AVRG WEEK	CPU PER KTRAN	PROD IMS PEAK WEEK	PROD IMS PEAK HOUR	PROD IMS AVRG WEEK	CPU PER KTRAN
JAN88	582	84.4	557	529	2.22	363	331	4.56	39.6	30.9	5.49	380	58.9	344	1.41
FEB88	594	86.1	566	531	2.23	402	358	4.60	41.5	32.4	5.52	389	60.3	352	1.42
MAR88	579	84	549	533	2.24	401	370	4.64	43.5	34	5.56	398	61.7	361	1.43
APR88	583	84.5	556	535	2.26	388	356	4.68	45.4	35.5	5.59	408	63.2	369	1.44
MAY88	587	85.1	559	537	2.27	378	348	4.72	47.4	37	5.62	417	64.7	378	1.45
JUN88	585	84.8	558	539	2.28	365	334	4.75	49.3	38.5	5.65	406	62.9	370	1.45
JUL88	589	85.5	559	541	2.29	372	333	4.79	51.3	40.1	5.69	435	67.5	398	1.46
AUG88	605	87.8	579	543	2.30	368	336	4.83	53.3	41.6	5.72	465	72	427	1.47
SEP88	616	89.3	584	545	2.32	374	342	4.87	55.2	43.1	5.75	494	76.6	455	1.48
OCT88	624	90.5	600	547	2.33	375	343	4.91	57.2	44.7	5.79	504	78.2	464	1.49
NOV88	626	90.8	591	549	2.34	371	340	4.96	59.1	46.2	5.82	514	79.7	473	1.50
DEC88	624	90.4	571	551	2.35	367	324	5.00	61.1	47.7	5.85	525	81.3	483	1.51

DAILY VOLUME PER WEEKDAY PRIME SHIFT

SAMPLE FORECAST
figure 8.

Track And Report. It is essential that the actual workload is tracked and reported routinely (figure 9). If the process is to improve, it must be generally understood. The better the parties involved understand the process, the better the quality of information for the next iteration. Don't be afraid to publish your batting average. It is probably better than anyone would have expected, and is sure to improve.

MONTHLY COMMERCIAL PERFORMANCE SUMMARY

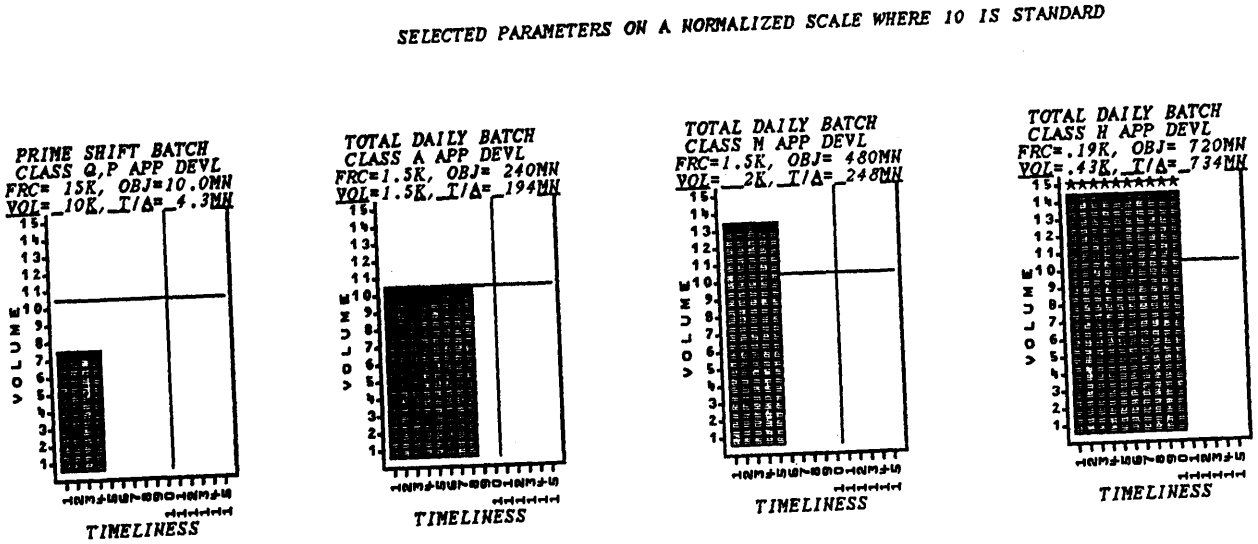
ACTUAL VALUES OF FORECASTED PARAMETERS -- AVERAGE OF 12-HOUR PRIME WORKDAY FOR INTERACTIVE, TOTAL WEEK FOR BATCH

INTERACTIVE	12-HOUR-PRIME-SHIFT								
	ACTUAL PRIME K-VOLUME	FORECSTD AVERAGE WEEK	ACTUAL PEAK WEEK	FORECSTD PEAK WEEK	RECORD PEAK WEEK	ACTUAL PEAK HOUR	FORECSTD PEAK HOUR	ACTUAL CPU PER K-VOLUME	FORECSTD CPU PER K-VOLUME
TSO	459.0	475.0	467.2	505.0	501.6	64.7	73.2	2.3	2.1
IMS	286.9	297.0	297.9	328.0	297.9	39.2	50.9	1.0	1.1
VSPC/APL	110.2	184.0	132.7	198.0	408.5	20.0		1.5	1.2
TSOE/APL	84.3	79.2	85.8	88.3	85.8	12.6		3.4	4.3
TSOE/DB2	16.7	21.7	19.1	27.8	19.8	3.6		5.9	5.3

BATCH	TOTAL-DAY						
	ACTUAL WEEKLY K-VOLUME	FORECSTD AVERAGE WEEK	ACTUAL PEAK WEEK	FORECSTD PEAK WEEK	RECORD PEAK WEEK	ACTUAL CPU PER K-VOLUME	FORECSTD CPU PER K-VOLUME
SCHED PROD	9.2	10.6	9.6	10.8	9.8	143.5	132.0
IMS ASPIC BMP	3.6	4.6	3.8	5.0	4.9	10.4	11.0
IMS OTHER BMP	3.2	3.6	3.5	3.8	4.0	250.8	165.0
APPL DEVLPMNT	24.9	27.7	26.4	30.5	29.5	43.6	19.6
OTHER	5.5	5.0	5.8	5.4	7.0	89.7	63.5

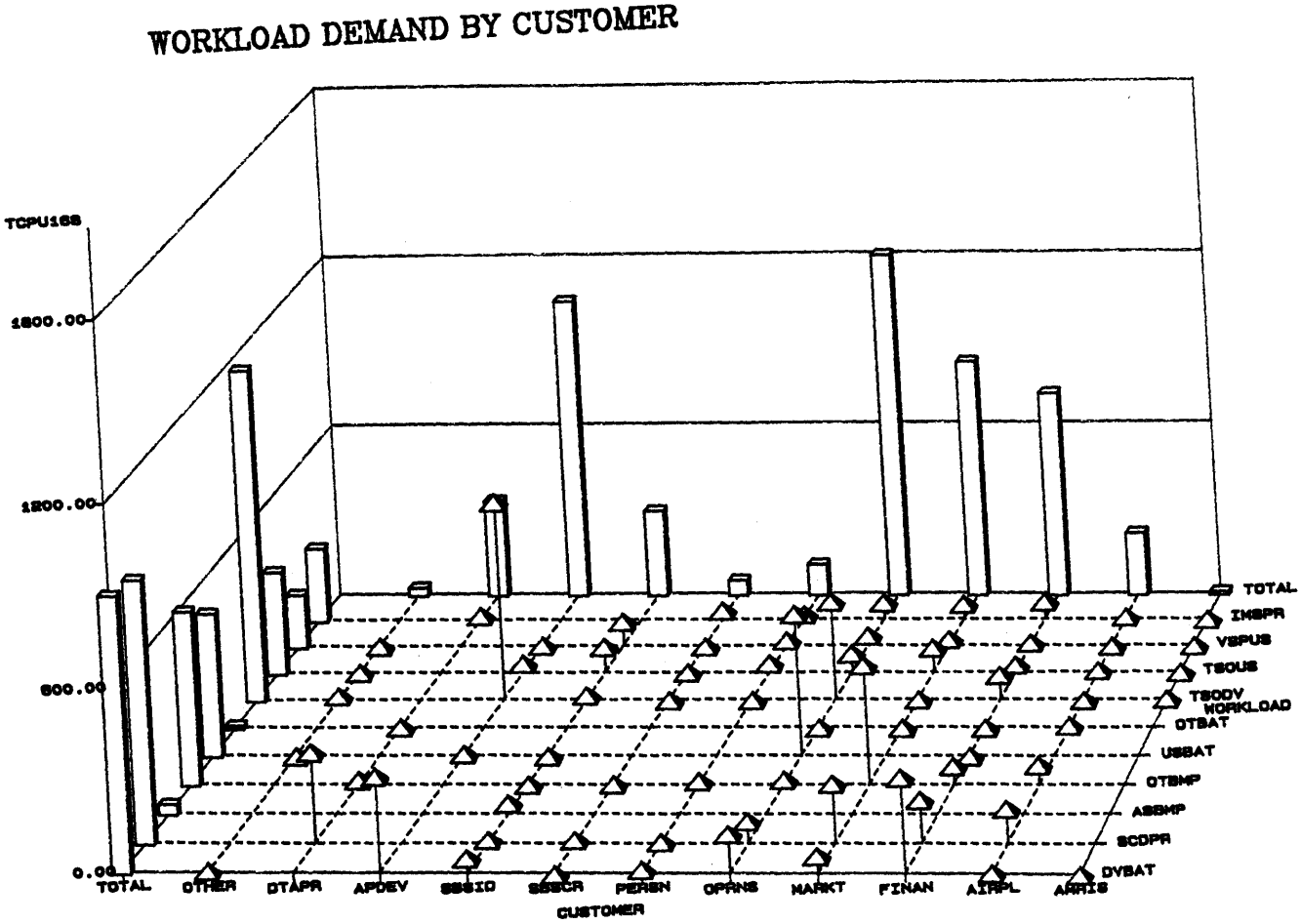
WEEKLY/MONTHLY ACTUAL VS FORECAST REPORT
figure 9.

The use of graphics helps the reader conceptualize information quickly. Figure 10 is a graphic (locally called a "Millergram") that normalizes both the forecasted volume and timeliness standard on a scale of 1 to 10 and allows a quick look at "how are we doing?"



COMPARE ACTUAL VS FORECAST WITH MILLERGRAMS
figure 10.

High level summarization of both workload and customer information can be displayed by SAS Proc G3D to give an insightive look at "who is doing what?" In figure 11, note that the vertical axis is in CPU units and the totals for each workload and customer are projected against the walls.



SHOW WORKLOAD DEMAND BY CUSTOMER WITH SAS PROC G3D
figure 11.

A FINAL WORD

This discussion here revolves around forecasting the workload. We are talking about the work to be done, not about the means for doing it. The discussion of a hardware/software environment to accommodate this workload and the costs of that environment are rightly the subject of a separate discussion. When both the work to be done and the cost of doing that work is made available to corporate management then the approval of the DP budget becomes a business decision. This is what we are here for.

Obviously, this brief paper does not comprise a "cookbook" for workload characterization and forecasting. It has simply attempted to suggest some approaches to this complex and challenging problem. There are a number of articles which provide more detail on specific aspects of the problem. A few are referenced here as a starting point.

SUGGESTED READINGS

[1] Artis, H. P., "Capacity Planning for MVS Computer Systems," CMG IX Processing (December 1978).

[2] Artis, H.P., "The Five Stages of Capacity Planning, "CMG '85 Proceedings (December 1985).

[3] Lo, T. L. and Elias, J.P., "Workload Forecasting Using NFU: A Capacity Planner's Perspective," CMG '86 Proceedings (December 1986).

[4] Miller, G. W., "Commercial Workload Demand Forecasting and Management Reporting at American Airlines," CMG '85 Proceedings (December 1985).

[5] Mullen, J. W., "Capacity Planning: Basic Elements for the Process," CMG '85 Proceedings (December 1985).

[6] Wong, F. A. and Pedriana, F.L., "Automated Forecasting of Large and Complex Workloads," CMG '86 Proceedings (December 1986).

APPENDIX A

RESOURCE IMPACT FORMS

The Commercial Resource Impact Forms are required to be completed as part of the feasibility study for a new application, and again as part of the detailed system design. They are intended to serve two purposes.

First they will be used to derive estimates of the cost of providing the application's resource requirements once implemented. Second, they will supply valuable advance information for the Commercial Demand Forecast used to support hardware planning.

These forms are intended to provide a convenient means of recording this information. Where possible, the questions are of the yes/no or fill in the blanks variety. If you have any difficulty with the question and require additional definitions or assistance, please referance the related pages which follow. If all else fails, call the Resource Measurement and Forecasting Group at 832-4631 or 832-4986 for assistance.

Some of the questions may be difficult to answer at feasibility time, although in all cases we would appreciate your best estimates. Questions which are optional at feasibility time are marked with an asterisk. By design time, most of the requested information should be available. Once provided, the information contained in these documents remains available to you for your examination and correction.

Once you have become familiar with these documents, it is hoped that you will not have any particular difficulty in using them. As you see ways in which they might be improved, please give us the benefit of your suggestions.

Good Luck!

RESOURCE IMPACT
PRELIMINARY DESIGN ESTIMATES
COMMERCIAL APPLICATIONS

PR/SR # _____
DATE _____

SYSTEM/APPLICATION NAME _____

DESCRIPTION _____
_____
_____
_____
_____
_____
_____

MANAGER _____ PHONE _____ MD _____
PROG/ANALYST _____ PHONE _____ MD _____

CUSTOMER/USER CONTACT _____ PHONE _____ MD _____
CUSTOMER/USER LOCATION _____

[ ] New application. Provide the name of an existing application to which this may be compared in size and function. _____

[ ] Modification to an existing application. Indicate the change in size and function which will result from this modification as a percent. _____
What was the original system name? _____

[ ] Both

What is the host environment for this project? Check ALL that apply.

[ ] Production Batch [ ] Production IMS [ ] TSOE/APL
[ ] IMS BMP [ ] IMS/DB2 [ ] TSOE/DB2
[ ] TERADATA [ ] Other _____

PLANNED IMPLEMENTATION DATE _____
PLANNED DESIGN REVIEW DATE _____

HELP/Definitions for page 1

PR/SR #: Project Request Number or Service Request Number. This procedure may be used for either.

DATE: Today's Date.

SYSTEM/APPLICATION NAME: Title or name by which this activity is commonly known.

DESCRIPTION: Brief narrative description of what is to be accomplished by this activity.

MANAGER: Name of the Applications Development and Programming Manager with primary responsibility for this activity.

PHONE: Business telephone number. MD: Mail drop or business address.

PROG/ANALYST: Name of the individual supplying this resource impact information (you).

CUSTOMER/USER CONTACT: Name of a customer or user contact for whom this activity is being accomplished.

CUSTOMER/USER LOCATION: Physical location of customer/user; city, state, and country. May reference existing generic locations such as reservations, airports, subscribers, etc.

New Application: This activity will create a new application.

Provide the name ....: To assist in the estimation of resource requirements for this activity, it would be helpful to relate it to an existing activity whose resource requirements can be determined. For example, "similar in function to XYZ, but one-half the size."

Modification to ....: This activity changes an existing application.

Indicate the change ....: Estimate the relative size of the application after the modifications are completed. For example, "will expand XYZ by 25%."

Production Batch: Batch jobs to be submitted by computer operations.

IMS/BMP: Batch Message Processing jobs run as batch jobs to access online IMS databases.

Production IMS: Utilizes interactive Message Processing Programs (MPP's) to access online IMS databases.

IMS/DB2: Utilizes DB2 to access online IMS or DB2 databases.

TERADATA: A Decision Enabling Technology application utilizing TERADATA equipment.

TSOE/APL: APL activity under the TSOE environment.

TSOE/DB2: A DB2 application under the TSOE environment.

Other: A host environment or combination of environments not specified above.

PLANNED IMPLEMENTATION DATE: Estimated date at which this activity will be implemented into a production status.

PLANNED DESIGN REVIEW DATE: Estimated date at which the next scheduled design review will take place.

NEW BATCH/BMP APPLICATIONS (If the answer to any of these questions is yes, go to page 3.)

1 YES NO Will this activity require IMS BMP batch jobs?

2 YES NO Will this activity require other batch jobs?

3 YES NO Will job submission be controlled by the user/customer?

4 YES NO Will more than 5 jobs be submitted per week?

5 YES NO Will job submission and processing normally need to occur during prime shift (6am to 6pm workdays)?

NEW INTERACTIVE APPLICATIONS (If the answer to any of these questions is yes, go to page 4.)

6 YES NO Will this application add new users to the system?

7 YES NO Will this application require the creation of any new transactions?

8 YES NO Will this application result in more than 100 transaction entries per day?

9 YES NO Will this application require the creation of new data bases?

PERIPHERAL REQUIREMENTS (If the answer to any of these questions is yes, go to page 5.)

10 YES NO Will this application require more than 5 tape mounts per week?

11 YES NO Will this application require the allocation of more than 200 megabytes (millions of bytes) of permanent DASD?

12 YES NO Will this application require the allocation of more than 700 megabytes (millions of bytes) of temporary DASD?

13 YES NO Will this application produce paper output requiring distribution by Document Control?

14 YES NO Will this application create microfiche?

1: Batch Message Processing (BMP) jobs are batch jobs which access online IMS databases.

2: Any batch job other than an IMS BMP job.

3: YES if any jobs are submitted by the user/customer, NO if all jobs are submitted by computer operations.

4: YES if the total number of jobs submitted for the week is more than 5 jobs, NO if otherwise.

5: YES if jobs must be submitted during the prime daytime hours, NO if the jobs may be run during evenings and on weekends.

6: YES if new individuals, terminals, or locations will result from this activity, NO if otherwise.

7: A transaction is here defined to be an interactive cycle involving a program of instructions. This cycle may be initiated by a user at a terminal or by another program, and will result in the return of information to a terminal, or the interaction with a database of information, or both. YES if a new program cycle is to be developed, NO if otherwise.

8: A transaction entry is here defined to be the initiation of a transaction cycle either by a user making an entry from a terminal or by a message switch. YES if the volume of this activity will exceed more than 100 entries per day, NO if otherwise.

9: YES if a new database will be created or if new elements are to be added to an existing database, NO if no new databases or elements will be created.

10: YES if the total weekly activity of this application will require more than 5 reel or cartridges of magnetic tape to be mounted, NO if otherwise.

11: YES if permanent datasets will require the allocation of more than 200 megabytes of DASD, NO if less than 200 megabytes. This question is intended to determine if a threshold of 200 megabytes of permanent DASD is to be exceeded. If so, you will be asked in a later section to estimate the total permanent DASD requirements in megabytes.

12: YES if temporary datasets will require the allocation of more than 700 megabytes of DASD, NO if less than 700 megabytes. For additional help in estimating temporary DASD requirements, see page 2b.

13: YES if documents will be printed on a paper media for distribution, NO if otherwise.

14: YES if microfiche output will result from this application, NO if otherwise.

ESTIMATE THE FOLLOWING TO THE BEST OF YOUR ABILITY

NEW BATCH/BMP APPLICATIONS

15 What percent of job submission will be under the control of the user/customer? _____

16 YES NO The volume of activity will vary seasonally.

17* if the answer to 16 was YES, then indicate those periods during which the peak demands will occur, the factors which will affect the volumes of activity, and the percent variation from high to low. _____
_____
_____

18 What number of jobs will occur during the peak week? _____

19* What percent of the job processing will need to occur during prime shift (6am to 6pm workdays?) _____

20* What is the estimated annual percent of growth in job volume for this application after implementation? _____

21* What is the basis for growth? _____
_____

HELP/Definitions for Page 3

15: 100% if alljobs are submitted by the user/customer, 0% if totally controlled by computer operations. If some of both, estimate the percent submitted by the user/customer.

16: The volume of activity for some applications is seen to vary in relation to season of the year, volume of passengers boarded, corporate budget cycle, employee vacation schedules, etc. Answer YES if there are significant (+ or - 10%) variations in volume of an activity from one time of year to another, NO if the activity is expected to be consistent throughout the year.

17: Identify the factors which will affect the variation in the volume of activity to the best of your knowledge.

18: The estimated total weekly volume of batch jobs to be submitted during a peak week of activity. If the volume of expected activity is not clear, you may wish to estimate the volume as small, medium or large, and then estimate what small medium or large would mean to you in terms of weekly volume of jobs.

19: Considerable flexibility in fully utilizing computer resources results from the ability to schedule production batch work outside of the prime hours of 6am to 6pm workdays. Answer 100% if all jobs must run during prime time, 0% if none must run during prime hours, or estimate the percent that must run during prime time.

20: Estimate the percent annual growth which will result after implementation. Consider new users and locations, planned expansion and normal growth.

21: Indicate the major factors which will affect growth, eg., new users or locations, planned expansion, volume of traffic, customer base, etc.

ESTIMATE THE FOLLOWING TO THE BEST OF YOUR ABILITY

NEW INTERACTIVE APPLICATIONS

22 YES NO The volume of activity will vary seasonally.

23* if the answer to 16 was YES, then indicate those periods during which the peak demands will occur, the factors which will affect the volumes of activity, and the percent variation from high to low. _____

_____

_____

24* How many new transactions will be developed? _____

25 What volume of transactions will occur during the peak day? _____

26* What percent of the transactions will need to occur during prime shift (6am to 6pm)? _____

27* During which hour of the day will the peak demand occur? _____

28* How many new users will be added to the system? _____

29* How many new terminals will be added to the system? _____

30* How many new locations will be added to the system? _____

31* What is the average input message size in bytes? _____

32* What is the average output message size in bytes? _____

33* What is the estimated annual percent of growth in transaction volume for this application after implementation? _____

34* What is the basis for growth? _____

_____

HELP/Definitions for Page 4

22: The volume of activity for some applications is seen to vary in relation to season of the year, volume of passengers boarded, corporate budget cycle, employee vacation schedules, etc. Answer YES if there are significant (+ or - 10%) variations in volume of an activity from one time of year to another, NO if the activity is expected to be consistent throughout the year.

23: Identify the factors which will affect the variation in the volume of activity to the best of your knowledge.

24: A transaction is here defined to be a interactive cycle involving a program of instructions. This cycle may be initiated by a user at a terminal or by another program, and will result in the return of information to a terminal, or the interaction with a database of information, or both. Estimate the number of new program cycles to be developed.

25: The estimated total daily volume of transaction entries to be submitted during a peak day of activity. If the volume of expected activity is not clear, you may wish to estimate the volume as small, medium or large, and then estimate what small medium or large would mean to you in terms of numbers of transactions.

26: 100% if all must occur during prime hours of 6am to 6pm workdays, 0% if none must occur during prime time, or estimate the percent that will occur during prime time.

27: Indicate the hour of the day when the peak level of activity is expected to occur. Use a number between 1 and 24 where hour 1 is the hour between midnight and 1am.

28: Estimate the number of new individuals who will be initiating system activity.

29: Estimate the number of new terminals that will be required.

30: Estimate the number of new physical locations from which activity will be initiated.

31: The input message size is the number of bytes of data entered at the terminal by the user/customer.

32: The output message size is the total number of bytes of data contained in the output screen returned to the terminal.

33: Estimate the percent annual growth which will result after implementation. Consider new users and locations, planned expansion and normal growth.

34: Indicate the major factors which will affect growth, eg., new users or locations, planned expansion, volume of traffic, customer base, etc.

ESTIMATE THE FOLLOWING TO THE BEST OF YOUR ABILITY

PERIPHERAL REQUIREMENTS

35* How many tape mounts will be required during the peak week? _____

36* What is the maximum number of concurrent tape drives allocated? _____

37 How many megabytes of new permanent DASD storage will be required? ___

38* How many megabytes of new temporary DASD storage will be required? ____

39* How many weekly lines of print will go to paper? _____

40* How many weekly microfiche masters will be produced? _____

Level 1 HELP/Definitions for Page 5

35: A mount occurs when a reel or cartridge of tape is loaded into a tape device. Estimate the total number of such mounts to be required by all jobs for the total week. If this may vary significantly over time, make estimates for a peak week. It may be useful to recall that a 3480 tape cartridge will contain approximately 200 megabytes of data.

36: The maximum number of physical tape devices which will be required at the same time. If this may vary over time, estimate a worst case.

37: The purpose of this question is to estimate the overall volume of permanent DASD which will be required by this application. While a more detailed breakout is not required, you may find it useful to detail your requirements to arrive at a total.

Data storage requiring the allocation of permanent DASD space are here defined as being one of three types: (Indicate space requirements in megabytes for each)

_____ Relational IMS or DB2 databases
_____ Generation Data Group (GDG) datasets
_____ Other non-relational non-GDG datasets (flat and VSAM files)

_____ Total for all Permanent DASD

It may be useful to recall that there are 630 megabytes of storage available on a single density 3380 device.

38: Temporary DASD space must be allocated for various work files and by sortout and sortwork requirements. Temporary space may be allocated and released at various times in the life of a job. It is of interest to know the largest concurrent requirement at any single time.

It may be useful to recall that there are 630 megabytes of storage available on a single density 3380 device.

39: The number of lines of print may be estimated as the number of pages for each report times the number of recipients times 50. If volumes may vary then estimate for a peak week.

40: A microfiche master is a sheet of microfiche film containing a maximum of 208 frames. A single document will require a microfiche master for each 208 pages of volume. Each single document will require at least one microfiche master even if its volume is as small as 1 page. Additional copies of such documents do not require additional masters. Estimate the number of microfiche masters to be produced. If volumes may vary then estimate for a peak week.

A STRATEGY FOR IMPROVING IO SUBSYSTEM DESIGN USING COST PER IO ANALYSIS

A. L. (Roy) Jones
Storage Technology Corporation

ABSTRACT

Traditional rules of thumb for IO subsystem design and maintenance are unsatisfactory. They often lead to the underutilization of very costly resources, less than desirable performance, and ever growing DP budgets. An alternative cost-effective approach is needed.

This paper outlines a strategy for improving the cost and performance of IO subsystems using a Cost-per-IO (CPIO) analysis methodology. By using the proper types of IO devices, empirically determined through CPIO analysis, most DP installations can both reduce costs and either increase workload throughput or decrease turnaround (response) time. CPIO analysis can easily be used to 1) evaluate current IO subsystem cost and performance, 2) assist data set assignment to an effective device, 3) assess the impact of alternative device assignments, and 4) estimate total system benefit if reconfigured. CPIO analysis will normally result in (suggest) a hierarchical IO subsystem design which places each individual data set on the storage device having the most appropriate related cost/performance characteristics. Thus, the cost and performance of the entire IO subsystem are improved. Operational and procedural benefits provide added value to most DP installations.

1.0 INTRODUCTION

In the current business world, Data Processing is often integral, if not critical, to the successful conduct of business by a corporation. Data Processing serves not only to enhance internal daily operations but also to provide timely and accurate support to valued corporate clients. Data Processing (DP) performance and related costs are becoming increasingly important to a modern corporation's bottom line profits.

Despite this, however, many DP installations today are faced with the somewhat unusual and highly undesirable situation of finding or being told that they cannot or should not fully utilize their DP hardware. For performance reasons, costly DASD storage devices are often only 50 to 70 percent utilized. Tape and DASD devices are often configured in "short" strings, thereby requiring a bloated number of even more costly device controllers. Recommended input/output (IO) channel utilization for the support of most device types is only 30 to 50 percent. Guidelines for maximum interval Central Processing Unit (CPU) utilization range from 70% to 90% with 70% the most commonly suggested for dynamic and highly visible online processing periods.

There are numerous studies available which employ queuing theory and industry findings to explain and justify limitations such as these. Hardware and software technology advances (e.g., dual-port or dynamic path reconnect) have been made over the years to relax restrictions but, despite these advances, the results are (or at least should be) unsatisfactory to a conscientious DP manager. Costly resources are underutilized and excess capacity must be provided to compensate.

1.1 COMMON PROBLEMS

Restrictive guidelines or rules of thumb may very well be valid for specific device types but, even when the common rules of thumb are followed, DP installations often experience slow batch job turnaround or extended periods of inadequate on-line system response. Problems which contribute to user dissatisfaction with DP service include:

- the low activity device syndrome
- high peak-to-average ratios
- workload and IO subsystem imbalance
- inability to fully utilize the CPU

The low activity device syndrome is characterized by the situation where low activity devices sharing a DASD string with high activity devices often suffer with degraded response time. Queuing theory explains this phenomenon: contention from the high activity device increases rotational miss probability for the low activity device and may therefore double or even triple the reconnect response time for an IO which is pending at the device. It may just be that the low activity IO is the real bottleneck or governing factor for online transaction response or batch job turnaround.

Interval peak-to-average ratios provide a well founded rationale for limiting overall average resource utilization to general guidelines during both online and batch processing periods, but they often do not fully account for the dynamic nature of the environment. In most cases, the guidelines are set at levels which will maintain resource availability for 85 to 95 percent of the processing period. It would be a rare DP environment, indeed, which would have a demand profile so consistent that 90 percent availability would also provide 99 to 100 percent availability. If resource demand significantly exceeds resource availability for only 5 minutes a day (approximately 1% of a prime shift), users can be severely impacted for 2 or 3 times as long while the demand backlog is worked off. It is during these times, however brief, that the phones of the Network Controller, Operations Manager, and even the Director of DP begin to ring with complaints from irate and frustrated users.

DP installations also tend to be characterized by workloads which shift their demand characteristics and resource requirements from period to period during the day, week, or month. Different types of IO devices are needed at different times. Once again, it would be an unusual DP shop which would have the luxury of a budget large enough to provide resources sufficient to meet all types of demand at all times. Resources are often shifted as needs and user pressures dictate, and inadequate "band-aid" measures are frequently used to provide adequate service for "special cases"; the end result is that some users are satisfied (more or less) at the expense of others.

The CPU is the most costly device in a DP installation but, even so, it often cannot be fully utilized. Peak-to-average ratios set guidelines for overall average CPU utilization at less than 100 percent. In addition, even when CPU demand is less than 100%, the demand cannot be fully serviced because the CPU must

wait for one or more IO operations to complete before proceeding (e.g., all tasks are waiting for data input). The latter condition is characterized on VMMAP or RMF reports by high queue lengths and the frequent occurrence of concurrent channel busy and CPU wait. The CPU has work pending but is waiting for IO.

The real problem which contributes to and actually aggravates the above problems, is the inability of the IO subsystem to respond to IO service requests in a timely fashion. More timely IO for both low activity devices and high activity data sets would permit the CPU to service processing demand more quickly. Speedier processing of individual tasks would reduce or eliminate demand backlog and tend to reduce the magnitude and volatility of peak-to-average ratios. Lower ratios would improve workload and IO subsystem imbalances. With faster IO, a CPU would experience less IO wait and could be more effectively and productively utilized during periods of peak processing demand.

1.2 SOLUTIONS: TRADITIONAL AND PROPOSED

Frequently suggested traditional solutions to DP performance problems include larger and faster CPUs, higher multiprogramming levels (MPLs), more main memory, more IO control units and/or storage devices, and more IO paths (or channels) to the IO devices [1]. All of these approaches involve the acquisition of more hardware resources at increasing cost and do little if anything to address the basic problems outlined above. With most of these solutions, device contention is aggravated and CPU demand is increased (to service the additional resources) while most job or transaction processing times are only slightly improved.

Traditional solutions tend to be implemented in piecemeal fashion when traditional rules of thumb indicate excessive utilization of (underutilized) devices. As such, these efforts tend to be preventative in nature (to prevent severe performance degradation) and do not offer significant performance improvement. All too often, the improvements which do occur are rather short lived.

The intent of this paper is to suggest an alternative way of thinking about IO subsystem design. The approach is designed to maximize or improve the utilization of current resources in a DP environment and to optimize the ratio of dollars spent to the amount of data processing work done. The hypothesis of this paper is as follows:

1) By making use of the proper set of IO devices, most DP installations can:

   - Improve device utilization
   - Reduce operating costs
     and
   - Improve performance
     (i.e., increase throughput
     and/or decrease turnaround)

2) The best device for any data set can be empirically determined through a process of Cost-Per-IO (i.e., CPIO) analysis

CPIO analysis offers the potential for an organization to improve DP performance while reducing related costs. CPIO analysis is based on a simple concept of producing the most work for the least cost through effective data storage device assignment and utilization. The proper application of CPIO analysis for IO subsystem design and maintenance will enable a DP organization to achieve what is, or at least should be, its primary objective --- improving the corporate return on dollars invested in Data Processing.

2.0 DEFINITIONS AND DEVICES

Before proceeding into development of the CPIO concept, the reader of this paper must understand the definition of certain terms used and the basic nature of the storage devices reviewed. The definitions establish a foundation and rationale for CPIO development. The IO devices are a tool and an end result of IO subsystem design; their fundamental characteristics establish guidelines for their effective utilization.

2.1 FUNDAMENTAL DEFINITIONS

Throughput is a measure of work units accomplished in a given time unit. Common DP measures of throughput include jobs per day, transactions per hour, and IOs per second. Turnaround is a measure of the time required to accomplish a unit of work. Common DP measures of turnaround include job run time, transaction response time, and device IO service time.

It is important to note that throughput and turnaround are not necessarily the inverse of each other --- the fact that an IO device provides an average service time of 20 milliseconds does not mean that the device can support 50 IOs per second. Path contention, queuing, and other factors come into play to prohibit linear inverse correlation of these measures.

While it is generally desirable to both increase throughput and decrease turnaround, the best overall measure of productivity in a DP shop is throughput. Simply put, this recognizes the fact that even though it may be desirable or even important to get work done in a timely fashion, it is even more important to just plain get the work done. Modern designs of Operating Systems, processing programs, and even computer hardware recognize this fact [2]; multi-tasking control programs and multi-processor CPUs all sacrifice turnaround (for the control of multiple operations) to achieve greater throughput.** Fortunately, greater throughput nearly always translates into a lower average turnaround over time and, hence, greater performance.

A work unit is a measure of work done. Common DP work units include jobs, jobsteps, transactions, and IO operations (IOs) to data storage devices. Among these, an IO is a basic building block; all of the other work units will invoke and require completion of a certain number of IOs on average. As such, improvements in IO processing will benefit the processing of all other common work units. For this reason, an IO is used as the basic work unit for development of the concept in this paper.

One additional definition is needed, not necessarily for development of the CPIO concept, but for clarification of what appears to be a common misconception in the DP industry. A megabyte (of storage device space) is a measure of space used, not work. A megabyte has occupancy or ownership cost only, relative to time. It is the use of (i.e., work done in) that time which establishes value. Work done to or from a storage device is an IO. A commonly heard statement that a certain storage device "should not be used because the cost per megabyte of storage is too high" is derived from misconception. Cost per megabyte (of storage space) on a device is not important; cost per IO (of throughput) on the device is.

** It is interesting to note that a rather loosely justified deviation from this philosophy is embodied in the introduction of synchronous expanded storage in the IBM 3090 series computer.

2.2 POTENTIAL DEVICES

Figure 1 presents a summary chart of commonly recognized current and pending technology storage device types. Nominal access time (order of magnitude) and the basic nature of data storage on the device are also depicted. It is beyond the scope of this paper to present a full discussion of each device, but certain characteristics are important to proper device selection [3,4] based upon (or in spite of) CPIO analysis results. The basic device considerations are described in the following paragraphs.

FIGURE 1

STORAGE HIERARCHY

Device Type	*Access Time*	*Nature*
CPU	nanoseconds	in-line
E/S	microseconds	on-line
SSD	milliseconds	on-line
CACHE	milliseconds	on-line
DASD	milliseconds	on-line
Optical Library	seconds	near-line
Cartridge Library	seconds	near-line
18-track tape, cartridge	minutes	off-line
9-track tape, buffered	minutes	off-line
Laser printer	-longer-	hardcopy
Impact printer	-longer yet-	hardcopy

The CPU (memory), while often not considered as such, is a storage device. Data accessed for manipulation by the CPU instruction processor is normally processed in chunks (words or frames) at CPU cycle time speeds. Even so, except in the very slowest processors, classical data records can be accessed in the nanosecond (billionth of a second) order of magnitude. CPU storage space is physically and logically (depending on the Operating System being used) very limited and therefore must be used very judiciously.

Expanded Storage (E/S), as currently implemented, requires an IBM 3090 CPU and extended architecture (XA). E/S currently supports paging only, so volatility (i.e., loss of data with loss of power) is not a significant issue. E/S cannot be shared between CPUs and plans for upgradeability (or transferability) to future generations of CPU can only be surmised; lack of upgradeability means that an investment in E/S must be written off over the life of the CPU to which it is attached. Access to E/S is by page frame synchronously (i.e., one page at a time without release of the central processor). IBM claims an access time of "about 75 microseconds" [5] but some industry measurements indicate that "the total time to resolve a page fault to expanded storage ---- is approximately 140 microseconds" [6].

Solid State Disk (SSD) offerings from several vendors, excluding IBM, include support for 2305, 3350, and 3380 device type interface. As such, SSD can be used on any CPU (and with any operating system) which supports one of these device types. SSD can be used for paging and non-paging applications, is sharable between different CPUs, and can be transferred from one CPU to another, even from multiple vendors. Access to SSD storage is asynchronous using 2 to 4 paths with an access time of 0.3 milliseconds (300 microseconds) plus channel transfer time for a usual total of 2-4 milliseconds. SSD storage is volatile but several technologies which eliminate or significantly reduce exposure to potential power loss are available in current offerings. These include battery backup, uninterruptible power supply (UPS), attached hard disk backup, and critical file duplexing via software. Volatility should certainly be considered for non-paging, non-temporary applications, but the issue should never preclude using an SSD if appropriate [7] for performance and cost improvements.

Cached DASD (CACHE) is simply rotating DASD such as 3350 or 3380 type devices enhanced by the inclusion of several megabytes of buffer storage (and the associated buffer management logic) within the rotating DASD control unit. The buffer does not represent permanent storage space; permanent storage continues to be the rotating DASD configuration. A cache control unit offers the potential for the significant improvement of access times to data stored on attached rotating DASD, but at significant additional cost for the acquisition and maintenance of the enhanced control unit. If properly [8] configured, a cache subsystem offers 6-10 millisecond service time and will permit significantly higher channel and device utilization than standard rotating DASD. A cache subsystem, however, is highly susceptible to misconfiguration; normally, data behind cache must exhibit a high (4+:1) read to write ratio and a suitable reference pattern in order to achieve cost effective beneficial results.

Rotating DASD is familiar to everyone. Rotating DASD potentially supports a wide range of IO per second with an equal or even larger range of IO service times. A nominally "good" service time can vary from 18 to 24 milliseconds. Device and operating system differences alter expected nominal results to some degree: dual port (or DLS) architecture increases supportable IO rate and/or decreases service time; dynamic path reconnect also helps. Data reference pattern has a noticeable impact on actual results with an individual DASD device. It is believed, however, that nominal results (or expectations) are valid for an entire DP installation. As a result, CPIO development uses a single value (from traditional guideline values) for rotating DASD of all types while recognizing that average values for any single environment might vary by 10 to 15 percent in either direction.

Automated optical or cartridge libraries are a device type which introduces a new concept of "near-line" storage. Data stored in such a library is not "on-line" with nearly instantaneous (millisecond) access but it is also not "off-line" with a manually initiated access which may take minutes or even longer. Mount times of 5 to 20 seconds are typical for library devices and may be very acceptable for many data sets currently maintained on relatively expensive rotating DASD. It may be time for DP installations to rethink archival policies in light of the potential offered by these devices. Users and programmers who keep large data files "on-line" simply to avoid the inconvenience of "off-line" storage may very well accept the minor delay associated with "near-line" storage, especially when the projected cost savings are substantial.

3.0 CPIO DEVELOPMENT

The basic objective of cost per IO analysis is to optimize (or improve) the overall cost and performance of data storage and retrieval subsystems within a Data Processing environment. The concept is based on the following tenets (developed and discussed at length in the preceding section):

1) Throughput is a better and more usable measure of overall DP productivity than turnaround. It is also more realistic for determining device and/or subsystem productivity.

2) A valid common measure of work done in a device, an IO subsystem, or a DP facility, is an IO operation. Average IO operations per second (IO/Sec) is the corresponding measure of throughput.

3) Improving the actual or potential throughput (IO/Sec) of an environment will improve the productivity of the environment. Improved productivity, with few exceptions, will result in improved turnaround (or performance).

4) Cost per IO (of throughput) is the ultimate measure of device or environment effectiveness. Improved effectiveness means:
   - lower cost with equal or greater productivity
   - higher productivity with equal or lower cost

5) Improving the effectiveness of the IO devices in an environment will improve the effectiveness of the total environment.

CPIO development is the process of deriving and understanding the cost per IO characteristics of various potential IO subsystem devices. Assumptions, results, and conclusions are presented in the following sections.

3.1 ASSUMPTIONS

In recognition of the myriad combinations of IO device variations, subsystem configurations, IO blocksizes, and IO device costs, several assumptions and scope limitations have been placed on this initial CPIO development. These limit the complexity of the development but provide a solid foundation for interpretation of the results and application of the concept.

To date, CPIO results have been developed for on-line devices only, in recognition of the fact that on-line IO subsystem effectiveness is usually the most important to a DP organization. Calculations and results have been developed for a representative "small" configuration and a representative "large" configuration of each "on-line" device type. Device throughput potentials (i.e., maximum IO per second) are based on a nominally "good" IO service time for the device when it is processing an average 4K to 6K average blocksize. The device characteristics used are summarized in Figure 2 and Figure 3. SPECIAL NOTE: CACHE configurations are simply a controller upgrade with an associated standard DASD configuration.

Cost calculations are based on list 5-year term lease payments (TLP) and maintenance from a standard vendor of the device type. It is fully recognized that some differences exist between vendors for the list price of equivalent configurations and that the list prices are usually discounted, but some basis for comparison must be used and this would seem to be as good as any. Costs associated with power, cooling, and floor space occupancy are excluded.

FIGURE 2

ONLINE DEVICES

Characteristics Used

	small config.	large config.
E/S	64 MB 3090-200	128 MB 3090-400
SSD	24 MB	768 MB
CACHE	6 MB sm. DASD	72 MB lg. DASD
DASD	2x4x 635 MB	2x16x 1270 MB

FIGURE 3

ONLINE DEVICES

Characteristics Used (cont.)

	nominal svc. time	maximum IO/sec
E/S	100 us	1000
SSD	3 ms	600
CACHE	8 ms	200
DASD	20-22 ms	40-50

List prices used for evaluation of the selected configurations are as follows:

Device Type	Config-uration	Purchase Price	Monthly 5-yr TLP	Monthly Maint.	Total $ Monthly
E/S	small	$405,000	$8,140	$500	$8,640
	large	695,000	13,970	900	14,870
SSD	small	128,500	2,281	641	2,922
	large	1,310,000	30,413	3,431	33,844
CACHE	small	202,425	4,256	864	5,120
	large	788,875	17,708	1,707	19,415
DASD	small	146,245	2,841	471	3,312
	large	467,895	9,940	1,116	11,056

Sources: IBM Service for Consultants, May, 1987
"IBM Credit Corporation Leasing Factors", Computer Price Watch, April, 1987
StorageTek Pricing Guide, April, 1987

The assumptions made for Expanded Storage deserve special comment. Access to E/S is synchronous (i.e., one page at a time without relinquishing the CPU). Access time reports vary from 75 to 140 microseconds. For simplicity, a value of 100 microseconds is used. At 100 microseconds per page (or IO), every 100 IOs per second consume 10 milliseconds (or one percent) of a processor per second (i.e., 100 IO/Sec = 1% of a 3090 processor). The author of this report is not aware of established or tested physical limits to E/S throughput but a practical limit of 1000 IO/Sec (10% of a processor) has been assumed; how much of the CPUs processing power should be realistically sacrificed in direct support of paging IO activity? It is also believed that CPU cost should be recognized and included as a factor (variable with IO rate) in the E/S cost per IO calculations, but, for simplicity and brevity, CPU costs associated with E/S block transfer activity have been excluded.

3.2 CALCULATIONS AND RESULTS

The calculation of cost per IO processed on a device involves the parameters of space, time, cost, and work. These parameters are variable by device type and by configuration size within device type. The basic CPIO analysis formula which incorporates each of these parameters is:

$$\frac{\text{COST}}{\text{IO}} \quad --- \quad \frac{\text{COST/SEC/MB}}{\text{IO/SEC/MB}}$$

COST/SEC/MB incorporates device configuration cost (COST) in relation to time (SEConds) and megabytes of storage space (MB); it is the occupancy cost for one megabyte of storage space in a device or IO subsystem configuration. IO/SEC/MB incorporates work done (IO) in relation to time (SEConds) and megabytes of storage space (MB); it is a measure of throughput per megabyte. COST/IO is the resulting measure of the effectiveness of a device or IO subsystem.

FIGURE 4

Cost Evaluation

Configuration Occupancy Cost($) : Cents

	$/Min	$/Sec/MB (x1000)
Small E/S	20.0	5.21
Large E/S	36.9	4.81
Small SSD	6.8	4.70
Large SSD	78.3	1.70
Sm. CACHE	11.9	.079
Lg. CACHE	44.9	.037
Sm. DASD	7.7	.051
Lg. DASD	25.6	.021

Figures 4 and 5 present initial development results for the reviewed device types and the assumed configuration sizes. Figure 4 develops occupancy costs and Figure 5 develops maximum utilization limits. The understanding and acceptance of these results is essential to the understanding and acceptance of the CPIO concept.

In Figure 4, $/Min represents the cost (presented in cents, or 0.01 dollars, for scaling reasons) of owning or having the corresponding configuration for one minute of time. It is simply the monthly cost of the device outlined in the preceding section divided by 720 hours and then divided by 60 minutes. For example:

Configuration: small DASD (2 directors + 4 3380-1x volumes)
Cost per month = 2841 (TLP) + 471 (maint.) = 3312 dollars
Cost per hour = cost per month / 720 = 4.60 dollars
Cost per minute = cost per hour / 60 = 0.0767 dollars
Cents($)/Minute = 7.7 (rounded)

$(cents)/minute results for the various configurations are highly variable. As might be expected, larger configurations of any device type cost more.

Also in Figure 4, $/Sec/MB represents the ownership or occupancy cost per second of one megabyte of storage space in the associated configuration. Once again, the results represent cents (0.01 dollars) and are multiplied by 1000; in other words, the results are really thousandths of a cent per second per megabyte (or they could alternatively be viewed as cents per second per gigabyte of storage space). For example:

Configuration: small DASD (2500 megabytes)
Cost per minute (from above) = 7.7 cents
Cost per second = cost per minute / 60 = 0.1283 cents
Cents/second/megabyte = cents per second / 2500 = 0.0000513
Cents($)/Sec/MB (x1000) = 0.051 (rounded)

$(cents)/Sec/MB has a definite decreasing trend as configuration size increases or the device type gets slower. This corresponds to the common (and frequently misapplied) recognition of this trend in relation to simple cost per megabyte of storage space.

In Figure 5, Maximum IO/Sec/MB is the computed maximum throughput capability per megabyte of each device configuration. It is simply the maximum throughput (IO/Sec) rate of the device (from Figure 3) divided by the total megabytes in the configuration (from Figure 2). For example:

Configuration: small DASD (2500 megabytes)
Maximum IO per second = 50 (from Fig. 3)
Maximum IO/Sec/MB = Max. IO per second / 2500 = 0.020

In reviewing Maximum IO/Sec/MB, it should be remembered that this value is not necessarily the absolute maximum rate supportable by the device; it is the maximum throughput rate which can be sustained on average while maintaining a nominally "good" average response time. As might be expected, the supportable IO/Sec/MB limit generally decreases as the devices get slower or the configurations get larger. One exception to the general trend is that a small SSD can support a greater throughput per megabyte than any size of Expanded Storage (mainly because E/S has a larger minimum size than SSD).

Minimum $/IO in Figure 5 essentially represents the maximum cost effectiveness for each configuration; therefore, the smaller the number, the better. Like

FIGURE 5

Cost Evaluation (cont.)

Maximum Utilization Limits
Cost($) : Cents

	Maximum IO/Sec/MB	Minimum $/IO(x1000)
Small E/S	15.6	.33
Large E/S	7.8	.62
Small SSD	25.0	.19
Large SSD	0.78	2.18
Sm. CACHE	0.08	.98
Lg. CACHE	0.01	3.70
Sm. DASD	0.020	2.55
Lg. DASD	0.0025	8.40

the other numbers above, this value is in cents (i.e., $ = cents) and is multiplied by 1000 for scaling purposes. This value is computed from the basic CPIO formula using the previously developed configuration residency cost and maximum throughput. For example:

Configuration: small DASD
$/Sec/MB (x1000) = 0.051 (from Figure 4)
IO/Sec/MB = 0.020 (from Figure 5)
$/IO (x1000) = 0.051 / 0.020 = 2.55 Cents per 1000 IOs

Because the maximum value is used for throughput (i.e., IO/Sec/MB), the result is the minimum cost per IO processed on the device if the configuration is used to its fullest potential. Cost per IO cannot be less without a an attendant degradation in average service time for the device.

3.3 INITIAL CONCLUSION AND EXTENDED RESULTS

The results presented above, particularly for minimum cost per IO, prove an important basic conclusion of CPIO development. Contrary to popular belief, fast devices such as CACHE and SSD, if properly utilized, can provide greater throughput at lower cost (per IO) than slower devices such as standard rotating DASD. Thus, effective use of faster devices offers the potential for a DP shop to improve performance and reduce cost of operation.

This conclusion, however, raises several attendant questions. How can effective use of a device be assured? What is the effect if the device is not used to its fullest potential? Can the best device for the placement of a data set be determined using CPIO results and, if so, can the financial and performance impact of placing the data set on an alternative device type be determined? Can the effectiveness of the current IO configurations at a DP installation be assessed? Can the potential for overall improvement to IO subsystem effectiveness be estimated?

In recognition of these questions, CPIO results have been extended to include an evaluation of cost per IO on the various device subsystems at throughput rates which are less than their maximum potential. Figure 5 gives a maximum throughput and a minimum CPIO for each evaluated configuration. When throughput is decreased from its maximum value for a specific configuration, the basic CPIO formula (i.e., Cost/IO = Cost/Sec/MB / IO/Sec/MB) produces an increasing CPIO.

Consider the following example of the impact of throughput variation:

Configuration: small DASD (2500 megabyte)
Max. throughput = 0.020 IO/Sec/MB
Min. cost per IO = 2.55 cents/IO (x1000)
Variable CPIO (x1000) = 0.051 / IO/Sec/MB

When this variable formula is used for a range of potential throughput rates, the results are:

Throughput(IO/Sec/MB)	CPIO(cents/IO x 1000)
0.020	2.55
0.010	5.10
0.001	51.00
0.0001	510.00
0.00001	5100.00

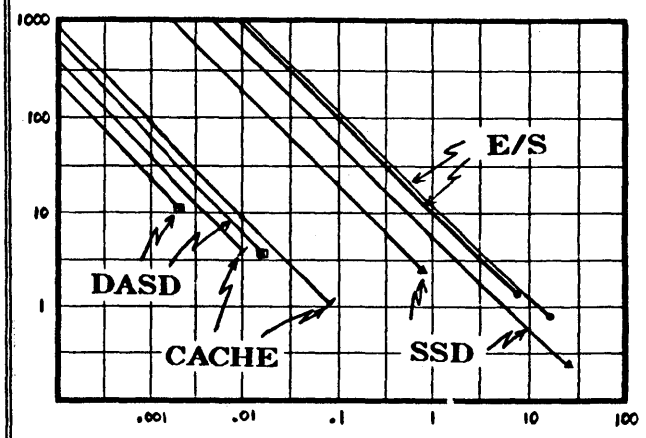
From these results, it can be seen quite readily that cost increases rapidly, in direct inverse proportion, as throughput decreases. It is also evident that any device which is significantly underutilized (e.g., a small DASD configuration processing less than 2.5 IO/second, or 0.001 IO/Sec/MB) can be very costly indeed.

Figure 6 presents variable CPIO results in graphic form for each device configuration within some reasonable parameter ranges.

FIGURE 6

Cost per IO Analysis Chart

Universal Scaling



Cost/IO vs. IO/Sec/MB by Device Type
X-axis: IO/Sec/MB (log scale)
Y-axis: Cents per 1000 IO Operations (log scale)

By using a logarithmic scale on each axis of the graph, the results for each configuration appear as a straight line. The delimiting point at the lower right end of each line is the maximum throughput, minimum CPIO value for the configuration. Each device type is represented by two lines; the rightmost line represents the small configuration for the device type and the leftmost line represents the large configuration. A large number of configuration sizes are available for each device type which would create lines (not depicted) at several points between and parallel to the extremes which are depicted. SPECIAL NOTE: the CACHE and DASD device configuration ranges overlap each other and create a special consideration which will be discussed later in this paper.

Since most of the potential cost values in Figure 6 are much higher than any reasonable or acceptable cost for a DP installation, Figure 7 is presented to put the CPIO values (Y-axis) into normal scale (i.e., not logarithmic). This presents a more usable form of the CPIO analysis data which will be referenced frequently from subsequent sections of this paper. It also accentuates the negative impact of inadequate throughput to a device and helps clarify lower limit throughput values for each configuration type.

The primary implication of this graph can be stated as follows:

> For each device type and configuration size, there is a limited range of throughput rates (IO/Sec/MB) which will provide an acceptable operating cost (i.e., cost per IO) and performance (i.e., IO service time).

For clarification of this basic CPIO concept, consider Figure 7 and a large DASD configuration (the leftmost curve). The results indicate that data placed on a large DASD configuration should exhibit an actual or demand average throughput rate between 0.0010 IO/Sec/MB (selected value) and 0.0025 IO/Sec/MB (max. limit from Figure 5).

Data operating at a rate less than 0.0010 IO/Sec/MB would have an undesirable operating cost (i.e., higher than 20 cents per 1000 IOs). Data operating at a rate greater than 0.0025 IO/Sec/MB would either exhibit subnormal performance (i.e., a service time longer than 22 milliseconds) or would require that compensating space on the device be excluded from effective utilization (e.g., 50 MB at 0.005 and 50 MB at 0.000 would maintain the supportable throughput level and the nominal service time).

If 20 cents per 1000 IOs is considered to be the maximum acceptable operating cost for an IO configuration, then desirable or target throughput rate ranges can be developed for each device type. The following table summarizes these results. The results are calculated but can be verified from Figure 7 (remember that the X-axis on the graph is logarithmic).

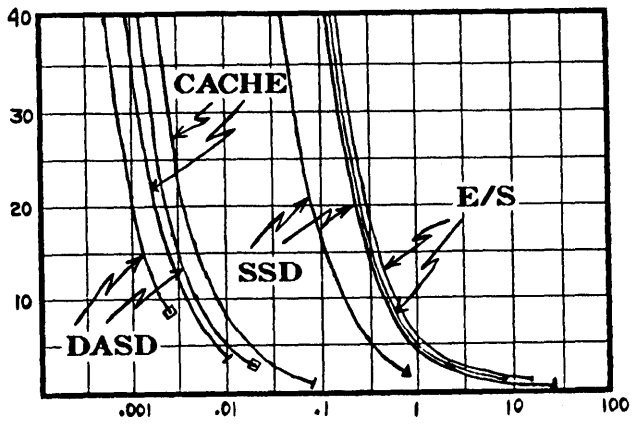
DEVICE TYPE	TARGET THROUGHPUT RATE (IO/Sec/MB) SMALL CONFIG.	LARGE CONFIG.	ALL CONFIG'S
DASD	0.0025-0.0200	0.0010-0.0025	0.001-0.020
CACHE	0.004-0.080	0.002-0.010	0.002-0.080
SSD	0.235-25.00	0.085-0.780	0.085-25.00
E/S	0.26-15.6	0.24-7.80	0.240-15.60

These target rates establish a clearly defined cost and performance effective throughput range for each on-line device type. They are an important and verifiable result of CPIO development and suggest a potential for improvement to IO subsystem design using CPIO analysis.

FIGURE 7

Cost per IO Analysis Chart

Limited Cost per IO Scaling



Cost/IO vs. IO/Sec/MB by Device Type
X-axis: IO/Sec/MB (log scale)
Y-axis: Cents per 1000 IO Operations

4.0 SUMMARY OBSERVATIONS

The CPIO development results presented above provide the foundation for several noteworthy observations. Items of interest include operating cost guidelines, device implications, and potential applications for the CPIO analysis concept.

4.1 COST GUIDELINES

A direct observation from the results is that the maximum utilization cost per IO for any on-line device type is less than 9 cents per 1000 IOs. If all devices in a DP shop are properly utilized at or near maximum IO rate and maximum space utilization, the DP shop should be able to operate at an average cost of 9 cents per 1000 IOs or less and maintain a nominally "good" average IO service time (e.g., 20-22 milliseconds for standard rotating DASD).

Many DP shops, however, cannot operate at maximum utilization levels on a constant basis. Because of workload fluctuations and the nonlinear degradation of device IO service times when recommended maximum throughput rates are exceeded, most shops must either maintain an average utilization less than maximum or suffer with less than desirable average IO service time. Operation at less than maximum utilization may be necessary but the reasons for doing so should be consciously recognized.

A nominal realistic operating cost target for any DP installation would appear to be 12 cents per 1000 IOs or less. If the most cost effective devices and configurations are properly employed in IO subsystem design, these results can be easily achieved even at 75% utilization or less.

4.2 DEVICE IMPLICATIONS

When the CPIO Analysis Chart for on-line device types (Figure 7) is studied, three gaps in the graph, relative to IO/Sec/MB, can be observed. An effective device type is not evident if the actual or demand throughput rate for data is:

- less than 0.001 IO/Sec/MB
- greater than 0.02 and less than about 0.10 (and the data is not cachable), or
- greater than 25 IO/Sec/MB

Gap 1 --- What should be done with data having a low throughput rate? Barring special circumstances such as a direct access file structure or a real need for immediate access, the traditional approach for low throughput data has been to place the data on tape. To put this in perspective, consider the following: a throughput rate of 0.001 IO/Sec/MB for a 200 MB data set equates to 17,280 IOs per day; this is equivalent to one pass of a 200 MB tape reel per day if the data is blocked at 12,000 bytes or only one pass every three days if the data is blocked at 4,000 bytes. Clearly, the data set is a classic candidate for placement on tape. CPIO results bear this out and indicate that cost per IO on disk would be at least 20 cents per 1000 IOs (unacceptable) and potentially much higher for a lower rate. The cost of maintaining the data set on disk would be at least $3.46 per day (i.e., 17,280 IO/day x $0.00020 $/IO) or over $100 per month; the cost for maintenance on tape would be the lower cost associated with 10 to 30 tape mounts per month.

Frequently, however, data sets with a throughput rate even lower than 0.001 IO/Sec/MB are (improperly) placed or left on DASD to avoid the inconvenience associated with manual tape mounts. This type of action is typically justified by placing the low activity data set on a volume with one or more very high activity data sets and claiming that the low activity data will simply utilize space left vacant and "wasted" for performance reasons. Thus, we have the crux of an important CPIO concept: CPIO analysis suggests that utilization of a DASD device to share both very high and very low activity data is normally an improper application of the DASD device on both counts. It is highly probable that the low activity data set would perform quite satisfactorily with lower cost on tape while the high activity data set(s) would perform much better and also with lower cost on a CACHE or SSD device. Recent developments with "near-line" storage devices would seem to make this even more attractive.

Gap 2 --- What should be done with data which has an intermediate throughput requirement (i.e., between 0.02 and 0.10 IO/Sec/MB) and is not cachable (i.e., will not provide assumed response time and throughput when cached)? Three current alternatives appear to be:

1) Relax cache requirements? This is a marginal possibility requiring individual case evaluation. Consideration of this possibility will reveal that doing so will only move the CACHE CPIO curves to the left and upward on Figure 7. As cache criteria are reduced, the CACHE curves will approach and actually rise above (i.e., become more costly than) the DASD curves; if improperly configured, CACHE could actually support less throughput [9] at greater cost than standard rotating DASD.

2) Place the data on standard DASD? This is always a potential course of action with recognition of higher (than CACHE) response time and the necessity to provide compensating "wasted" space on the receiving device or suffer with even worse than normal DASD response time.

3) Place the data on SSD? This can also be done. Response time would be better than for CACHE and much better than for DASD but the cost per IO would be significantly higher.

Each of these alternatives should be evaluated in light of the DP facility's objectives and priorities. If the data set is used primarily for batch processing, DASD may be the preferable option. It should be noted, however, that many installations today are being sold (improperly, we think) on the option to "relax cache requirements". If the data set is used primarily for on-line processing from terminals, the SSD option can do wonders for transaction response time and user good will, and may very well be worth the incremental cost.

A fourth possibility for the intermediate gap is that a new device type may be needed in the industry. Except for those relatively few data sets which are cachable, there is no effective device for the support of data sets in the intermediate throughput range. And many of these data sets are well known to DP shops; they tend to be those troublesome data sets which place the greatest stress on the total throughput capabilities of a standard DASD subsystem and which require constant and expensive tuning effort.

Gap 3 --- What about throughput demand greater than 25 IO/Sec/MB? Such demand is relatively rare but does exist. The most obvious device for placement of this data, at least while being used, is real CPU memory; operating systems and processing programs usually account for high usage and, by internal design (e.g., paging and buffering), provide for central processor residence of the highly active data. IO devices of the future may provide for extensions to this external device limit, but there are none in evidence today.

4.3 APPLICATION POTENTIALS

A final observation regarding the CPIO development results is that the concept offers significant potential for improving the effectiveness of IO subsystem design. The CPIO Analysis Chart (Figure 7), in particular, offers a simple framework which can easily be used to:

- evaluate current IO subsystem effectiveness
- assist identification of the best device for data set placement
- assess the impact of alternative device assignment, and/or
- estimate the cost and performance benefits offered by a suggested reconfiguration

The chart incorporates both cost and performance characteristics, permitting improvement of both. By assigning each data set to the device which is most appropriate to its throughput, usage characteristics, and importance, the overall effectiveness of an entire IO subsystem can be improved or optimized.

5.0 THE OPTIMIZATION PROCESS

IO subsystem design and maintenance is the process of assigning DP information files to IO devices, sizing and installing the appropriate equipment, and maintaining the resulting configuration. CPIO analysis offers the potential for a DP shop to truly "optimize" IO subsystem design if optimization is viewed as getting greater benefit for less expenditure.

In common practice today, files (or data sets) are assigned to data storage devices based on qualitative criteria (e.g., usage or file structure) rather than quantitative criteria (e.g., cost per IO). The number of device types employed tend to be rather limited. Standard configurations include devices for hardcopy,

off-line, and on-line data storage but there is often just one device type in each category. Multiple offerings within each category tend to be available only if prompted by sheer need (e.g., intolerable performance).

CPIO analysis provides a quantitative technique to assist the IO subsystem design process. CPIO development results indicate that several different types of devices are necessary to effectively support data sets with different throughput demands; CPIO analysis enables and enforces consideration of multiple device types.

Figure 8 lists the component steps of an IO subsystem reconfiguration effort. CPIO analysis will accomplish or augment the first five of these steps. While the concept can be used with exact configuration cost and throughput calculations for exact results, the CPIO Analysis Chart (Figure 7) can be easily used for working approximations and meaningful results. Applications of the chart are discussed below.

FIGURE 8

The Optimization Process

1. **Identify or verify need**
2. **Establish best device per data set**
3. **Evaluate data set placement(s)**
4. **Total system requirements by device and establish configuration levels**
5. **Project revised cost & service time**
6. **Review**
7. **Implement**
8. **Review**

SPECIAL NOTE: It is recommended that all data used for CPIO analysis (e.g., RMF, SMF, VMMAP, etc.) cover at least a representative 24-hour interval --- longer if possible.

5.1 STEP 1 - IDENTIFY OR VERIFY NEED

In the DP industry, hardware configuration changes are rarely scheduled or even contemplated without a present or imminently pending need. The need may be a simple application requirement for a new feature or more storage space, but more frequently it is a need growing out of performance concerns (e.g., complaints from users or measured deteriorating service levels). Performance and application needs will always occur (without respect to CPIO analysis).

CPIO analysis introduces the possibility for a new type of need, or measurable cause for concern --- namely, installation effectiveness. Quantifiable potential improvements to performance and cost imply a need (to improve operating effectiveness). Quantifiable shortcomings in either cost or performance with respect to other installations should also imply a need (to correct operating deficiencies).

CPIO analysis provides a method to measure current effectiveness and to assess the potential for improvement. The process of current installation assessment using CPIO analysis can be outlined as follows:

1) By device type or subsystem set (i.e., a configuration grouping within device type such as long strings versus short strings), determine:
   - total IO per second (average)
   - average configuration size
   - total megabytes in the set

2) For each set, compute the IO/Sec/MB; divide the total IO per second of the set by the total megabytes in the set.

3) On the graph in Figure 7, for each set:
   - move right on the X-axis the the computed IO/Sec/MB
   - move straight up from this point to intersect the curve which corresponds to the average configuration type and size established

4) The corresponding Y-axis value denotes the cost per IO for the device type or set; if more than one set is present at the installation, repeat steps 1 through 3 for each and compute a weighted average of the total environment CPIO.

5) Use standard RMF or VMMAP data (in DASD device activity reports) to determine weighted average IO service time for the installation.

Given the cost per IO and average IO service time, thus determined, we have a complete measure of current installation effectiveness. Each installation must be its own judge as to the need or desire for improvement (refer to Section 6 for comparative industry findings).

5.2 STEP 2 - OPTIMIZE DATA SET DEVICES

Once a need for reconfiguration is established, CPIO analysis can be used to easily identify the most cost effective potential devices for one or more data sets. The word "potential" is stressed because the the indicated device may not be functionally appropriate for the data set (see Step 3) or it may not be available (or provideable) in the required configuration size at the completion of configuration sizing (see Step 4). CPIO analysis is a design process which may involve reiteration between steps 2 and 4.

In step 2, the first activity is to determine the current or needed throughput (IO/Sec/MB) of the data set. Current throughput will produce acceptable results but consideration must be given to hidden or potential needs if the result is to be optimal. Consider the following:

- do multiple copies of the file exist?
- is there excess unused space in the allocated space?
- what will the throughput be if the data set is on a faster device?

With the established throughput, move across the CPIO Analysis Chart's X-axis to the determined rate and move vertically to the first curve line intersected. The line which is intersected is associated with the most appropriate and cost effective device with one possible exception --- if the line is the large CACHE line, either CACHE or an "intermediate" size DASD could be

considered. Step 3 will evaluate this situation; if the data set is cachable, then CACHE would be the device of choice because of the superior performance offered.

5.3 STEP 3 - EVALUATE DATA SET PLACEMENT

After the best potential device for a data set has been identified, the indicated placement must be evaluated with respect to proper compatibility of the device and the data set characteristics. For example, a data set which appears to be most appropriate for CACHE may not exhibit the read to write ratio and locality of reference necessary for the effective utilization of cache. If it does not, an alternative device assignment must be determined.

The CPIO Analysis Chart can be used to readily assess the impact of an alternative data set placement. The point of curve intersection established in Step 2 can be moved up to a new device-type curve with an increased performance and cost per IO or it can be moved to the left to the curve for a device offering less performance at equal or greater cost. The intersection point cannot logically be moved down or to the right: since the point was established by the first curve moving up, there is nothing lower; moving to the right would imply improved throughput, but maximum throughput demand has already (properly) been considered.

The relative cost and performance values for alternative devices, thus determined, can be subjectively evaluated to select a final recommended placement. Considerations of "cost vs. performance" have always existed but CPIO analysis puts the cost factor into meaningful quantitative terms.

5.4 STEP 4 - SUMMARIZE REQUIREMENTS

After all data sets have been assigned a "recommended" device type, appropriate device configuration sizes must be established. Thus far, we have a list of data sets, each requiring a certain number of megabytes of storage space and imposing a certain demand throughput on a set of recommended devices. The next step is to total (add up) the IO/Sec and the megabytes required for each device type; from these two values, the total IO/Sec/MB for the device type can be computed by simple division.

The total current throughput and space requirement, thus determined, provide a foundation for sizing an appropriate configuration of the suggested device. In addition to these current values, however, anticipated future growth, previous device assignment compromises, and the potential for revised utilization of the whole computer system (due to implied performance improvements), should now be considered.

The CPIO Analysis Chart can easily be used to facilitate this entire process. Cost per IO of the summarized configuration for each device type can be easily estimated. The potential impact of providing excess capacity for future growth can be assessed. The impact of possible changes to overall average throughput rates can be readily determined. The result of this process should be a complete, fully understood, and comfortable IO subsystem design which will probably incorporate several device types and configuration sizes in a clearly hierarchical structure.

5.5 STEP 5 - ESTIMATE BENEFITS

The last step is to estimate, in quantitative terms, the potential benefit of implementing the suggested IO configuration design. It is recognized that any change to a hardware configuration can be very traumatic for a DP installation, and a suggested reconfiguration which probably implies acquisition of new device types with some associated procedural changes can be even more frightening. The benefits which are offered must be clearly understood before a change will be warranted and approved.

The acquisition of new devices implies new expenses --- so where are the potential cost savings? This question pinpoints a basic and fundamental effect of using CPIO analysis for IO subsystem design and reconfiguration. By moving data sets with higher throughput demand to more appropriate (faster) devices, the space occupied by the data sets (and probably much more compensatory "wasted" space) is freed on the original device(s). If the impact of high throughput data sets is removed from low throughput devices, the low throughput devices can be utilized much more effectively (i.e., with a higher percentage of space allocation). The storage space, thusly reclaimed, has a definite monetary value which can be recovered by disposition (e.g., sale) of the vacated devices or recognition of the value in deferred future growth expense (i.e., the space can be reserved for future growth so that acquisition of new devices will not be required later). Performance improvement is a natural consequence of matching data set throughput demand with device throughput capability.

The CPIO Analysis Chart (Figure 7) provides a simple framework to estimate the new cost per IO on each recommended device type configuration. Device characteristics (Figure 3) provide the target estimated service time for each device type. For simplicity, the median value can be used. With these parameters for each device subsystem (designated by subscript $_n$), total system cost per IO and average IO service time can be estimated.

Total system cost per IO (X) can be computed as a weighted average using the following formulas:

$$A_n = (IO/Sec)_n * (Cost/IO)_n$$
$$B_n = (IO/Sec)_n$$
$$X = (A_1 + A_2 + ...) / (B_1 + B_2 + ...)$$

Total system average service time (Y) can be estimated with a similar weighted average development:

A_n = the average service time for device type $_n$
(E/S=0.1ms, SSD=3ms, CACHE=8ms, and DASD=20ms)

$$B_n = (IO/Sec)_n * A_n$$
$$C_n = (IO/Sec)_n$$
$$X = (B_1 + B_2 + ...) / (C_1 + C_2 + ...)$$

Thus, we have an estimate of the primary reconfiguration benefits. The cost per IO and average service time values provide a measure of revised installation effectiveness.

Figure 9 presents a list of somewhat intangible but meaningful added benefits. Several discussions presented above have laid a foundation for an understanding of the listed benefits so they will not be expanded in detail. It should be noted, however, that perhaps the most important added value benefit of a suggested reconfiguration is embodied in the higher throughput potential of the faster devices. The space on the faster devices (e.g., CACHE or SSD) may be fairly well utilized, but often, the throughput potential of the device will not be stressed. Available throughput headroom on the device will enable the device to easily respond to peak demand surges of short duration and provide the significant final benefit listed --- namely, improved overall response time consistency.

Needless to say, all of these benefits should be considered when the merits of a suggested device reconfiguration are under evaluation. The financial, operational, and procedural advantages are substantial.

FIGURE 9

Other Benefits

- Reduced floor space
- Reduced power & cooling cost
- Low activity IO much faster
- Less CPU with constant work
- Simplified cabling/operation
- Homogeneous DASD tuning
- Fewer channels required?
- Room for workload growth
- More consistent IO response

6.0 EXAMPLES AND SAMPLE INDUSTRY FINDINGS

Examples and industry findings which are presented in this section are based on and derived from actual Data Processing installations. Most of these installations are customers of Storage Technology Corporation (StorageTek) and utilize various types and sizes of StorageTek equipment. By and large, they did not make use of on-line device types other than standard rotating DASD at the time of initial CPIO analysis.

6.1 EXAMPLE 1 - PROVIDING FOR GROWTH

Figure 10 summarizes the environment, study findings, and primary results for a medium size (3081 class CPU) installation. The condition which prompted analysis of this installation was the pending implementation of new applications which forecast the need for 20 to 25 percent additional on-line storage space and a corresponding increase in total system IO activity.

Contemplated expansion, using traditional methods, implied the acquisition of 10 to 15 gigabytes of new DASD storage at considerable additional cost. The comparative value (i.e., list) cost of the current configuration was $37,980 per month with a potential addition of $6,470 to $8,767 per month for a DASD control unit and 2 or 3 3380-E class (i.e., double capacity) storage units. Cost per IO would initially increase but would return to approximately 16.7 cents per 1000 IOs after the addition of anticipated new activity.

CPIO analysis presented tne opportunity to actually lower the monthly cost to a comparative $33,369 while including preassigned space for the projected growth. The resulting average cost per IO would begin at 14.7 but would decline to approximately 12 cents per 1000 IOs as the new applications were added and the throughput increased.

6.2 EXAMPLE 2 - A CASE FOR COST REDUCTION

Figure 11 summarizes the environment, preliminary study findings, and estimated results for a fairly large

FIGURE 10

Example 1

Installation Overview

- DASD Environment and Device Configuration:
  - 80 8380-1x volumes (68.3% allocated)
  - all devices in 2x16 strings
  - 6 strings in total; 2 shared with VM & ACP
- IO activity:
  - 37.8 million IO's in 5 days (87.5 IO/second)
  - avg. blocksize = 5413; avg. response = 20.9 ms

Study Findings

- 11.9% of total system IO would fit in 48 MB of SSD storage (.20 IO/sec/MB)
- 16.0% of total system IO could be placed on four 8380-2x volumes behind 12 MB of cache and yield a modelled 5.9 ms host response (.0025 IO/sec/MB)
- Remaining data could be placed on 32 8380-2x volumes configured as two 2x16 strings with an estimated 19 ms IO response (.0014 IO/sec/MB)

Summary Results

- Pre-reconfiguration (little room for growth):
  - total cost/month = $37,980; cost/IO = 16.7
  - avg. IO service time = 20.9 ms
- Post-reconfiguration (room for 20-25% growth):
  - total cost/month = $33,369; cost/IO = 14.7 (12)
  - avg. IO service time = 15.0 ms

(multiple 3090 class CPUs) installation. This CPIO analysis was performed simply as a service adjunct to a customer requested tape subsystem analysis. The results, however, were quite surprising when put into the perspective of operating cost for the entire installation over a year's period.

The CPIO analysis was conducted for the DASD subpool and activity for one mainframe only. Whereas the results for Example 1 were based on extensive customer supplied data including SMF, RMF, VMMAP, and even GTF trace, these results are estimated and extrapolated from SMF data only. It has been repeatedly observed by this author that total DASD IO activity (including system data sets not reported by SMF) exceeds SMF reported activity by a factor of 10 to 15 percent. Adjustments for this additional throughput have been factored into the analysis results which are presented here.

The basic conclusion of CPIO analysis for this DP installation was that nearly $400,000 per year in comparative cost could potentially be saved for the on-line storage subpool of a single mainframe. The potential cost savings for this entire installation, even if the estimates are incorrect by quite a bit, could amount to one million dollars ($1,000,000) per year or more.

The major observation and point of this example is that while cost per IO differentials of a few cents per thousand IOs may seem insignificant, they can produce surprisingly large totals over a year. Savings of the magnitude indicated here should have a meaningful impact on any corporation's bottom line profits.

FIGURE 11

Example 2

Installation Overview

- DASD Environment and Device Configuration:
  - 140 3380-1x volumes (60% allocation (?))
  - two 2x16, four 2x12, three 2x8, and nine 2x4 (7.8 avg. string length) - 7 channels used
- IO activity:
  - 34.5 million IO's in 2 days (199.4 IO/second)
  - avg. blocksize = 4105; avg. response unknown

Preliminary Study Findings

- 10% of total system IO would fit in 12 MB of SSD storage (1.5 IO/sec/MB)
- A statistically estimated 27.5% of total system IO could be placed on 8 8380-2x volumes behind 18 MB of cache and yield an estimated 8 ms host response (.005 IO/sec/MB)
- Remaining data could be placed on 56 8380-2x volumes configured as two 2x16 strings and two 2x12 strings with an estimated 20 ms host response (.0016 IO/sec/MB)

Estimated Results

- Pre-reconfiguration:
  - equiv. cost/month = $83,892; avg. cost/IO = 16.2
  - avg. IO service time = ???
- Post-reconfiguration:
  - total cost/month = $50,684; avg. cost/IO = 9.8
  - avg. IO service time = 15 ms

6.3 SAMPLE INDUSTRY FINDINGS

Figure 12 presents a table of CPIO analysis findings for several existing DP installations (identified by letter only). Installation A is the same as that which was reviewed in Example 1; the others represent different environments which range in size from quite small (i.e., 4341 class CPU) to quite large (i.e., multiple 3090 class CPUs).

The data at the top of Figure 12 represents the results of initial CPIO installation assessments while the data at the bottom (i.e., below the horizontal separating line) represents the estimated results of several reconfigurations suggested by detail CPIO analysis. Reconfiguration estimates are not given for those installations which could not (or would not) provide adequate supporting data. An RMF or VMMAP report covering 24 hours or longer can be used for installation assessment; device assignment, however, requires detail information regarding individual data set size, throughput, and usage. A source for detail information (e.g., SMF or IO trace data) is not readily available in many DP shops.

Observations which can be extracted from an inspection of Figure 12 include the following:

1) As a general rule, the higher the cost per IO, the lower the average IO service time (i.e., milliseconds per IO); there are some obvious deviations from this trend, but it is beyond the scope of this paper to analyze the variations in detail

FIGURE 12

Sample Industry Findings

Installation	*Average Cost per IO*	*Average ms per IO*
A	16.7	20.9
B	15.8	23.6
C	14.4	26.7
D	12.6	26.8
E	11.3	36.0
F	9.7	24.0
G	9.6	35.7
-------- *With Reconfiguration* --------		
A	12.0	15.0
B	11.5	14.2
D	10.6	15.4
E	11.3	9.0
F	8.2	11.5

2) If initial cost per IO is relatively high, reconfiguration with CPIO analysis tends to offer significantly lower cost and moderately improved service time

3) If initial cost per IO is relatively low, reconfiguration with CPIO analysis tends to offer moderately lower cost and significantly improved service time

4) In all cases except one, CPIO analysis improves both cost per IO and average service time; for the exception, cost per IO remains the same but service time improves by 75 percent --- a seemingly worthwhile tradeoff

Figure 12 offers a gauge against which the results of any new CPIO installation assessment can be compared. The results in Figure 12, along with others which have been collected, suggest that there is a "normal" price/performance curve for traditional IO subsystem designs. New installation assessment values can be readily compared with the "normal" curve for the identification of significant deviations.

The results in Figure 12 also suggest a different curve for the price/performance of IO subsystems designed with CPIO analysis. The magnitude of potential movement from current values to the "CPIO" curve can be used to make a preliminary estimate (i.e., without detail CPIO analysis) of the benefit which might be derived from detail CPIO analysis.

7.0 SUMMARY AND CONCLUSION

The concept of CPIO analysis which is presented in this paper is suggested as a strategy for improving the IO subsystem design and maintenance process. It is not intended to replace traditional methods for assigning new application data sets to appropriate devices; in most cases, demand throughput needs are not well

understood prior to implementation of a new application or system, and traditional methods are quite adequate. It is proposed, however, to augment traditional methods and assist data set reassignment or system redesign when real throughput characteristics become known or change over time.

Despite the many potential benefits offered by CPIO analysis, it is not realistic to believe that many DP installations could immediately adopt a suggested reconfiguration. CPIO analysis offers quantifiable performance and financial improvements. It also offers a number of operational and procedural advantages to most installations. These should at least warrant recognition of the suggested reconfiguration and be used to establish guidelines for future equipment acquisition. Over time, the projected improvements to installation effectiveness (reduced cost and improved performance) would be achieved.

REFERENCES

[1] Artis, H.P., Buzen, J.P., "IBM MVS I/O Configuration Management", a Technology Transfer Institute seminar, November 5-7, 1986

[2] Singh, Y., King, G.M., Anderson, J.W., "IBM 3090 performance: A balanced system approach", IBM Systems Journal, Vol 25, No 1, 1986

[3] Cheung, D., "Storage Strategy", CMG '86 Conference Proceedings, December, 1986

[4] Nievas, E., "Storage Systems Solutions: A Management Perspective", CMG '86 Conference Proceedings, December, 1986

[5] Tucker, S.G., "The IBM 3090 system: An overview", IBM Systems Journal, Vol 25, No 1, 1986

[6] Buzen, J.P., "IBM 3090-200 Architecture and Performance", a paper from BGS Systems, April 24, 1986

[7] Moore, F., "Solid State Device Considerations", CMG '85 Conference Proceedings, December, 1985

[8] Mungal, A.G., "Cached I/O Subsystems: Analysis and Performance", CMG '86 Conference Proceedings, December, 1986

[9] Hanna, C.S., "Degradation of IMS Performance Using 3880-23 Cache Controllers", CMG '86 Conference Proceedings, December, 1986

REPORT ON THE LARGE SYSTEMS STORAGE HEIRARCHY

Fred Moore and Jerry Percell
Storage Technology Corporation

ABSTRACT

Driven by conversions from batch processing to online systems and the quest for improved cost-effective storage solutions, the storage heirarchy has added several new levels in the 1980s. At the beginning of the decade, disk, tape and various mass storage devices were available. Today, through improvements in architecture and technology, these basic storage levels have expanded to include high-performance or solid-state disk, cache control units, automated cartridge systems and buffered tape controllers. This report will discuss the new storage heirarchy and explore likely future developments in each storage level.

INTRODUCTION

For years, data processing was perceived as a backroom activity within a company's finance organization. Today, data processing has changed significantly and become a major strategic asset of the enterprise. In fact, data processing has evolved into the critical role of information management. The degree of efficiency with which a company manages its information has a direct impact on the success of the business. During the 1980s, we have witnessed major conversions of batch applications to online systems required to implement the information management architecture of the enterprise. As more and more aspects of the business become automated, the corporate goals for productivity and efficiency become tied directly to the information management functions. The age of information is upon us. (See Figure 1)

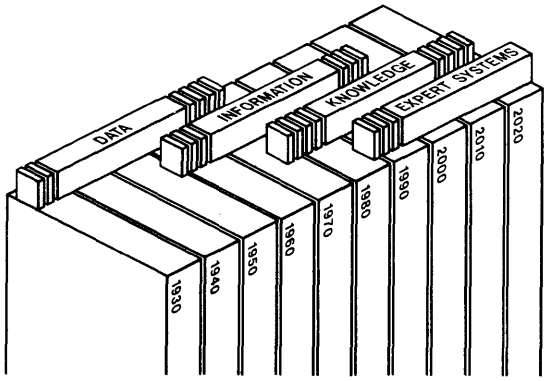


Figure 1. Data Processing Era

More specifically, in the IBM and compatible large systems environment, a primary concern of information management has become the issue of how to store and retrieve information cost effectively. In the 1980s, storage and retrieval systems have evolved from their 1970 counterparts to provide several potential solutions for information storage and retrieval. Those offerings may be depicted in five general levels of storage each having unique properties of performance, capacity and price to address storage. (See Figure 2)

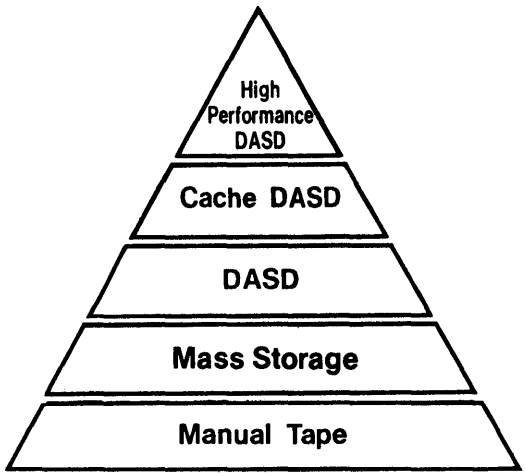


Figure 2. Information Storage Heirarchy

In particular, this report will discuss this new storage hierarchy describing architecture, performance, practical applications and future considerations for each storage level.

HIGH-PERFORMANCE DASD

ARCHITECTURE

During the early 1970s, paging operating systems were announced for use on computers implementing the virtual storage concept. In addition to processor and I/O constraints, these systems introduced a new bottleneck for performance analysts — paging. It was not until 1979 that the first high-performance (also called solid state) disk was delivered to address this problem. High-performance disk drives typically consist of one, two or four control modules, a variable amount of RAM (random access memory), and a variety of options to help address loss of power such as battery backup, internal hard disks and UPS (uninterruptible power supplies). The flexibility of microcode has permitted a high-performance disk to appear to the operating system as 3380 and 3350 disk images, and the 2305-2 fixed head drum. Several capacity and addressing options are available to permit configuration flexibility. Today, 3380 is the most common image used. Pathing options for high-performance disk provide one, two or four concurrent I/O operations per subsystem. (See Figure 3) From an architectural view, solid-state devices have relatively few mechanical components resulting in relatively high subsystem availability, a critical requirement for today's online response-oriented systems.

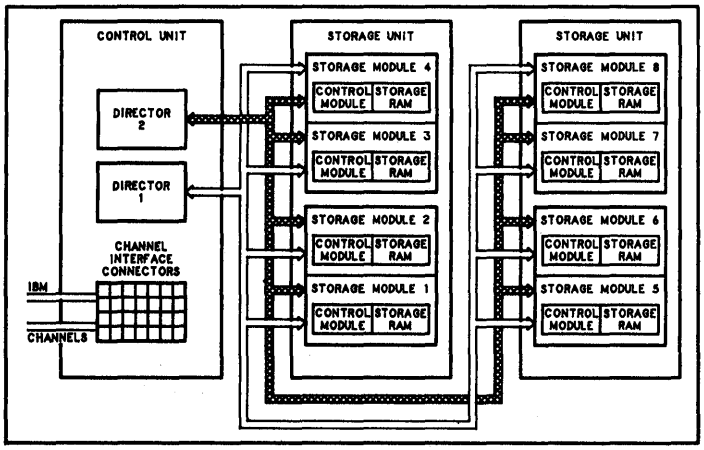


Figure 3. Configuration of SSD

Each storage module looks like an individual device to the host computer

Control Modules –
Provide control unit functions and channel interfaces sharable among CPUs.

RAM Storage –
High density RAM providing from 12 megabytes to two gigabytes of storage. Typically configured as several 3380 disk images.

Volatility Support –
Optional features to protect data in case of loss of power including batteries, UPS and small winchester disks.

PERFORMANCE

The use of RAM as the storage medium for high-performance disk provides just that — high performance. Access of data is provided typically of 0.3 milliseconds (ms) before data transfer begins. High-performance DASD sits atop the storage hierarchy because of its ability to provide consistent I/O response times. Unlike rotating DASD, these devices have no actuator arms to move (SEEK), no media to rotate (latency) and don't release the host channel to locate a record. This results in erratic response times for non-cache DASD. The 0.3 ms access time results in typical 2 to 4 ms I/O response times, depending on the blocksize, and between 300 and 400 I/O requests per second per I/O path are commonly attainable. In addition, the elimination of the RPS factor of rotating DASD and any data staging overhead of cached DASD provide for channel utilization to exceed 70 percent busy before any significant channel queuing occurs. Though not a requirement, high-performance disks usually reside on dedicated channels.

Performance Summary

- *0.3 ms access time*
- *Asynchronous I/O, 2 to 4 paths*
- *300 to 400 I/O requests per I/O path*
- *70 percent + channel utilization*
- *Sharable high-performance I/O between CPUs*
- *Eliminates RPS miss*

The initial and primary use of high-performance DASD was for the paging and swapping requirement of virtual operating systems such as MVS and VM. Until 1984 these devices were offered only as the 2305-2 fixed head drum with 11.25 megabytes (MB) of storage per 2305-2 image and only 90 MB capacity per storage subsystem. In 1984, the first 3380 device images were introduced and the era of non-paging applications officially had begun. These device images immediately addressed a broad range of response-time-starved applications for online information processing. The provision for device images that mirrored rotating DASD may have been the most significant enhancement for high-performance DASD. A new image facilitated the portability of data between real 3380-class DASD and high-performance 3380 without the need to convert blocksizes or change space allocations via JCL (job control language). This level of storage address applications requiring 100 percent consistent response, either random or sequential access modes and READ, WRITE or UPDATE processing modes. These include:

High-Performance DASD Applications

- *Response-critical load libraries*
- *High-performance data bases*
- *Work areas, storage pools, scratch pad areas*
- *JES2 checkpoint*
- *Catalogs*
- *Indices*
- *Security systems control data sets*
- *Enqueue/Dequeue files*
- *Sort work areas*
- *ACP/TPF Agent Assembly Area (AAA) files, file pool and fixed files*
- *Paging/swapping*

What about the initial use of high-performance DASD — paging and swapping? Page and swap are still the most common uses of these devices today though the growth of non-paging is accelerating quickly. The 31-bit addressing capability of MVS/XA architecture was a major extension from the 24-bit addressing capabilities of MVS/370. This 31-bit addressing permits an address space to be 128 times larger in MVS/XA than MVS/370. **(See Figure 4, Figure 5)**

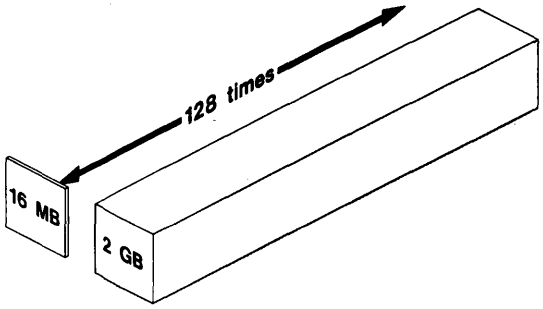


Figure 4. MVS/XA Virtual Address Space

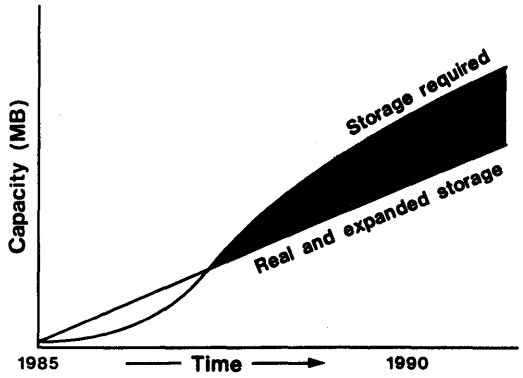


Figure 5. Virtual Storage Requirements

It is estimated that the 31-bit address space is being consumed at the rate of about one bit per year and that by 1992 the limits of this architecture will be pressed. It is also unlikely that real memory and expanded storage will be able to address this growth cost effectively, necessitating a continued requirement for paging. To support this view, IBM announced in October 1986, support for up to 256 page/swap data sets, up from a previous limit of 64, for MVS/XA 2.2.2. These data sets address a page/swap requirement above and beyond the capabilities of real and expanded storage. The additional page/swap data sets also may have uses yet to be announced such as for certain high-performance data files or dedicated functions belonging to specific address spaces.

FUTURES

High-performance DASD is emerging as a general purpose solution for response-critical requirements. By providing 100 percent space utilization and over 70 percent channel utilization, efficiency of architecture has become very appealing as a cost-effective benefit of this device. The issue of volatility has become better understood as users realize that any storage device has an exposure to data loss. For example, power outage can cause loss of data on high-performance DASD; head crashes or media damage can cause loss of data for rotating DASD; and edge, stretch, tear and humidity can cause magnetic tape data to be non-retrievable. Though battery backup, hard disks and UPS may *address* part of the volatile nature of these devices, the *solution* is to keep a secondary copy of any data on systems-accessible storage, i.e., accessible by a different channel path, control unit and device for maximum data availability. Advances in RAM technology permit a continued increase in device capacity. RAM has evolved from one kilobit chips in 1970 to one megabit chips in 1986 with continued development promising even higher capacity subsystems. Since the physical definitions of the device image are established by microcode, future changes in track size, cylinder size and device type can be changed relatively easily compared to other storage devices. Additional advances will permit higher availability and throughput with quad and octal porting capabilities exploiting the path connection limits of MVS/XA design. High-performance DASD also should provide an indicator of futuristic online storage devices by signaling an early beginning of the end of RPS based architectures. A summary of future advancements for high-performance DASD includes:

High-Performance DASD Futures

- *Improved connectivity and availability*
- *I/O channel utilizations 90 to 95 percent range*
- *Non-volatile storage*
- *Larger capacity*
- *Asynchronous I/O in conjunction with DASD*
- *Duplex capabilities*
- *Faster I/O channel transfer rates*

CACHE DASD

ARCHITECTURE

The use of cache DASD controllers has gained acceptance steadily throughout the 1980s. In the past few years there have been major advances in processor speed while the performance of rotating DASD has seen only small improvements. Cache controllers reduce I/O service times for many I/O requests, thus improving overall I/O subsystem performance. By storing data that is frequently accessed in the cache memory, the I/O request is satisfied by accessing data from the cache as opposed to rotating DASD. The cache then serves as a high-performance holding area for frequently referenced DASD data. The cache itself is RAM storage and represents an enhancement to the DASD control unit. Most cache designs today include two storage directors (control units), a variable amount cache, and microprocessor engines to manage the cache contents independent from the directors. This provides increased parallelism by permitting concurrent channel or device operations to occur with cache operations. **(See Figure 6)**

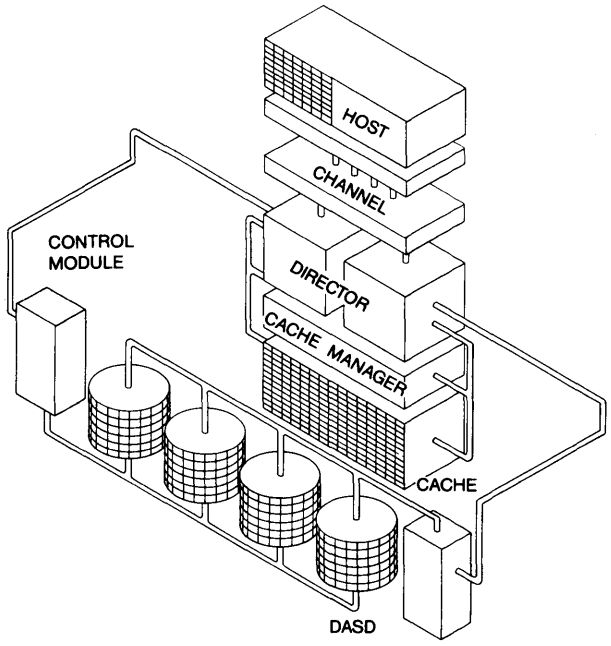


Figure 6.

PERFORMANCE

Cache provides the capability to anticipate host processor demand and manage the flow of data according to each program by directing operations to or from disk and cache. The higher the percentage of requests satisfied from the cache, the greater the performance improvement. Today's cache control units only benefit from high READ hit ratios as WRITES must always be directed to the DASD volume for data integrity purposes. Thus, for WRITES, a Device End interrupt is not posted until the data is written successfully on rotating DASD elongating I/O response times for the WRITE. A high percentage of READS compared to WRITES is most important for today's cache implementation. Most DASD volumes today demonstrate good caching characteristics. They include:

- Relatively high I/O rates
- High READ/WRITE ratios (at least 2:1)
- High data re-use (subsequent cache hits)
- A locality of reference (random or sequential access within)

The higher the percentage of I/O requests satisfied by the cache (hit ratio) the better or more consistent the performance. Devices that don't meet these characteristics actually can degrade performance. Figure 7 shows a DASD subsystem model using generalized trace facility (GTF) channel command word (CCW) trace. Note that volumes 302 and 307 actually degrade performance when placed behind the cache due to a high percentage of WRITES. Cache is therefore subject to misconfiguration and analysis of DASD volumes to be cached or not cached must occur normally. When properly configured, a cache subsystem will provide more consistent response for the attached DASD subsystem (six - 10 ms service time) and can improve channel utilization above the typical 35 percent busy threshold for non-cache DASD. Depending on the READ hit ratio, which eliminates RPS for the READS, channel utilizations can reach 60 percent or more.

370 APE PROGRAM VERSION = V5.10 SYBERCACHE ANALYSIS 00 04/18/86 11:12:22

DEVICE MODELED = 8380 CHANNEL (IN KBS) = 3000

SUMMARY OF MODEL PREDICTIONS OF PERFORMANCE

CACHE SIZE

	NO CACHE			1.5 MEG				3.0 MEG				6.0 MEG				12.0 MEG				18.0 MEG			
DEV ADD	HOST RESP (MS)	DEV RESP (MS)	%DEV UTL	HOST RESP (MS)	DEV RESP (MS)	%DEV UTL	% IMP	HOST RESP (MS)	DEV RESP (MS)	%DEV UTL	% IMP	HOST RESP (MS)	DEV RESP (MS)	%DEV UTL	% IMP	HOST RESP (MS)	DEV RESP (MS)	%DEV UTL	% IMP	HOST RESP (MS)	DEV RESP (MS)	%DEV UTL	% IMP
300	18.7	18.7	1	14.4	14.2	1	23	13.8	13.6	1	26	13.3	13.2	1	29	12.7	12.6	0	32	12.6	12.5	0	33
301	21.9	19.6	17	14.8	12.1	14	32	14.4	11.9	14	34	14.1	11.6	14	36	13.8	11.4	13	37	13.7	11.3	13	38
302	17.0	16.7	3	18.6	18.2	3	-10	18.3	17.9	3	- 8	18.0	17.6	3	- 6	17.7	17.3	3	- 4	17.6	17.3	3	- 4
303	19.6	17.8	15	8.7	6.7	10	56	6.8	5.4	8	65	5.0	4.1	5	71	3.6	3.1	4	82	3.0	2.7	3	84
304	16.9	16.1	8	15.3	14.1	8	9	14.7	13.6	8	13	14.3	13.3	8	15	14.0	13.0	7	17	13.8	12.9	7	18
305	23.8	22.2	12	20.6	18.6	11	14	19.8	18.0	11	17	19.2	17.6	10	19	18.8	17.2	10	21	18.6	17.0	10	22
306	19.9	19.6	3	19.2	18.3	4	4	18.2	17.4	4	8	17.2	16.4	4	14	16.2	15.5	3	19	15.6	15.0	3	21
307	16.7	16.5	2	17.0	16.7	2	- 2	16.8	16.5	2	- 1	16.6	16.3	2	1	16.4	16.1	2	2	16.3	16.0	2	2
308	15.5	15.5	0	7.0	6.9	0	55	6.8	6.7	0	56	6.6	6.5	0	57	6.5	6.4	0	58	6.2	6.1	0	60
309	20.1	20.0	0	19.9	19.7	0	1	19.7	19.5	0	2	19.3	19.2	0	4	19.2	19.1	0	4	19.1	19.0	0	5
30A	17.6	17.6	0	15.9	15.8	0	10	15.4	15.3	0	13	14.9	14.9	0	15	14.4	14.3	0	18	14.3	14.3	0	19
30B	14.5	14.5	0	15.9	15.8	0	-10	15.7	15.6	0	- 8	15.4	15.3	0	- 6	15.3	15.2	0	- 5	15.2	15.1	0	- 5
30C	15.8	15.7	0	10.5	10.3	0	34	10.0	9.9	0	36	9.9	9.7	0	38	9.6	9.5	0	39	9.5	9.4	0	40
30D	15.4	15.4	0	16.0	15.9	0	- 4	15.7	15.6	0	- 2	15.4	15.3	0	0	15.2	15.1	0	1	15.2	15.1	0	1
30E	21.4	21.0	4	16.7	16.0	4	22	16.0	15.4	3	26	15.4	14.9	3	28	14.8	14.4	3	31	14.6	14.1	3	32
30F	18.7	18.5	1	9.0	8.8	1	52	7.7	7.5	1	59	6.6	6.4	1	65	6.1	6.0	1	67	6.0	5.8	1	68
WT AVE	20.2	18.8	4	14.6	12.9	4	28	13.8	12.3	3	32	13.0	11.6	3	36	12.3	11.2	3	39	12.1	11.0	3	40

Figure 7. Summary of Model Predictions of Performance

APPLICATIONS

The popularity of cache has grown steadily in the 1980's in conjunction with the growth in online and response critical applications. It is important to understand the characterization of DASD volumes to attain maximum cache benefit. DASD volumes with a high percentage of WRITES compared to READS would be bypassed from caching in today's cached architectures. Studies show that several applications benefit from cache and most often represent the good cache characteristics mentioned in the section on performance. These include:

Cache DASD Applications

- *Partitioned data sets (libraries)*
- *Data base indices*
- *Certain data bases*
- *Sequential data sets*
- *TSO user data sets*
- *Pageable Link Pack Area (PLPA)*
- *Catalogs with high read percent*

FUTURES

Today's cache architectures are somewhat dependent on the user's ability to configure a cache subsystem carefully. Misconfiguration can actually lead to reduced performance. Future cache controllers will begin to remove this responsibility from the user by placing more intelligence into the cache management function via software and microcode. The near-term enhancement is one that permits the cache to monitor good or poor performance and change cache algorithms accordingly. Without user intervention, therefore, the cache becomes more self tuning. The use of DEFINE EXTENT and LOCATE channel command words is the first step in this direction. These channel command words are used throughout many MVS/XA access methods, program products and utilities providing the intent of the channel program: i.e., READ or WRITE, and LOCATION information in advance of the actual I/O operations. Knowledge of this information in advance can provide valuable 'clues' to the cache control unit enabling functions such as anticipatory staging/de-staging and advanced purge of un-needed tracks or blocks. Exploitation of these commands through host software will enable more data to become a candidate for cache. Another future function for cache controllers is non-retentive WRITE. Non-retentive WRITE causes the WRITE I/O operation to post a device-end interrupt when the data is written into the cache.

Today, cache DASD subsystems don't post device end until the data has been successfully written on the back-end DASD. This results in a significant performance penalty for WRITES. The ability to cache WRITES without a performance penalty permits more data to become cacheable and can provide performance improvement for certain high READ- and high WRITE- intensive data. Some cache control units permit the user to use a software command called BIND to retain highly active data elements in cache at all times and permit the cache to approach the performance of high-performance DASD for this data. Further use of BIND and the implementation of non-retentive WRITES will require cache sizes to increase beyond the 64 to 72 megabyte cache sizes available today. Future directions for cache control units include:

Cache Futures

- *More intelligent cache management in control unit*
- *Further use of host software — less user envolvement*
- *Non-retentive writes (cache WRITES without penalty)*
- *Microprocessors providing parallelism for front-end and back-end separation*
- *Larger cache sizes*
- *Increased connectivity*
- *Faster I/O channel transfer rates*

DASD

ARCHITECTURE

The use of DASD as the primary online storage device continues to grow at over 35 percent annually. Coupling adequate performance with high capacity and non-volatile storage, DASD satisfies both online and batch requirements for information storage and retrieval. DASD also has the ability to overwrite old data with new data without any need for a separate erase pass. Today's DASD most often uses thin-film heads and magnetic oxide coated aluminum substrate media. The devices use a flying head non-contact recording method to avoid head media wear at the high rpm (typically 3,600 rpm) speeds. Present DASD drives fly the head at about 10 microinches above the disk surface. Ultra-smooth sputtered thin-film media has demonstrated glide testing capabilities at four to five microinches though this capability remains in the laboratory. Further lowering of the flying height is improbable as the rules of aerodynamics begin to fail as the flying height approaches the dimensions of the air molecules themselves. The thickness of the magnetic layer on media also has been reduced over time as flying heights are lowered and areal density increases. **(See Figures 8 and 9)**

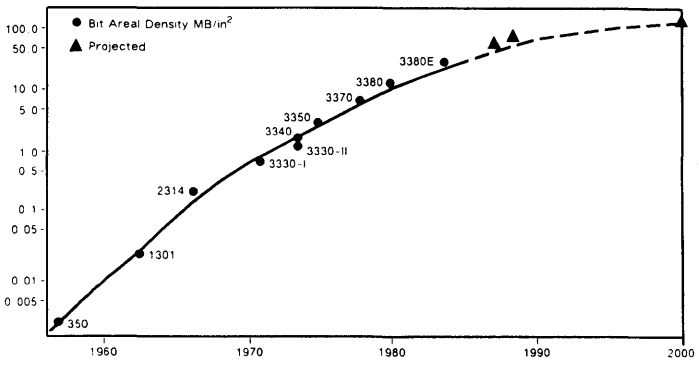


Figure 8. Rigid Disk Areal Density Trends

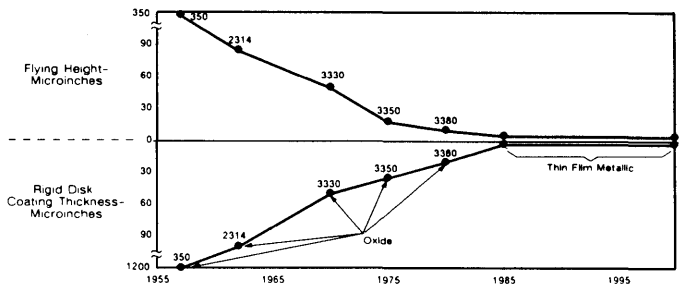


Figure 9. Flying Height Magnetic Media Coating Thickness

Ultra-low flying heights require a sealed and ultra-clean atmosphere as the slightest bit of microscopic debris will cause a head crash. The media used for DASD in the IBM large computer market has a diameter ranging from 10.5 to 14 inches with capacities per actuator (access mechanism) at 1.26 gigabytes.

PERFORMANCE

DASD represents the storage architecture of maximum surprise when it comes to I/O response time. SEEK, latency, data transfer and RPS miss are the major response components of a DASD I/O and all can vary as the device and I/O paths get busier. Unlike high-performance DASD where the RPS factor is zero, every RPS miss adds another 16.7 ms to DASD I/O response time. As path busy approaches 35 percent, queuing theory indicates that one full DASD revolution of 16.7 ms (at 3,600 rpm) will occur for every successful I/O. Dual ported I/O was introduced in the 1970s and significantly helps the RPS miss issue. In addition to dual port, dynamic path reconnect also helps reduce RPS miss by allowing the I/O operations from the device to reconnect to the host via a path other than the one on which it was initiated. This major I/O architectural change is available through the MVS/XA operating system and was the major I/O change from MVS/370. The following figure illustrates performance comparisons between high-performance DASD, cache DASD and non-cache DASD subsystems.

Performance Comparison

	Chan Busy Max Percent	I/O Rate (4K)/Sec	Initial Service Time
High Performance DASD			
	70 - 75	*700 - 800*	*1 - 3 ms*
Cache DASD			
	50 - 60	*200 - 300*	*5 - 20 ms*
DASD			
	35 - 45	*90 - 100*	*20 - 50 ms*

Another revealing measurement of DASD performance is access density or the I/O rate/second/gigabyte of storage. This index has declined steadily since 1970. On a per unit of storage basis, the unit of storage has less performance capability with each new device introduction. Performance per DASD actuator has not kept up with the increased capacity per actuator, in effect 'weakening' the I/O capabilities of the device per unit of storage. A strong case can be made that the practical limits of capacity per actuator are being reached calling for more actuators per unit of storage in future DASD.

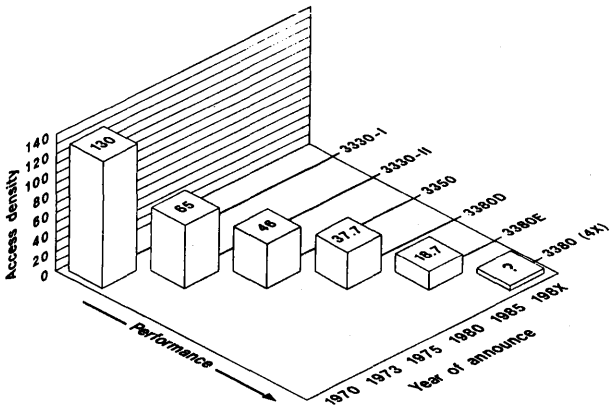


Figure 10.

Access Density

Year	Device	Subsystem Capacity	I/O Sec	Access Density
1970	*3330-1*	*400 MB*	*52*	*130*
1973	*3330-11*	*800 MB*	*52*	*65*
1975	*3350*	*1,270 MB*	*58.4*	*46*
1980	*3380(1X)*	*2,570 MB*	*95*	*37.7*
1985	*3380(2X)*	*5,040 MB*	*94*	*18.7*

Note: Subsystem includes two channel paths, two control units and four actuators.

APPLICATIONS

DASD is the most versatile of all storage devices permitting a non-volatile, relatively good performance for random, sequential, READ, WRITE and UPDATE uses. A GUIDE survey published in late 1984 indicated that the amount of allocated space per actuator was declining steadily. The access density decline of subsequent technologies has driven users to allocate less space on the DASD to reduce contention and maintain acceptable performance levels. (See Figure 11)

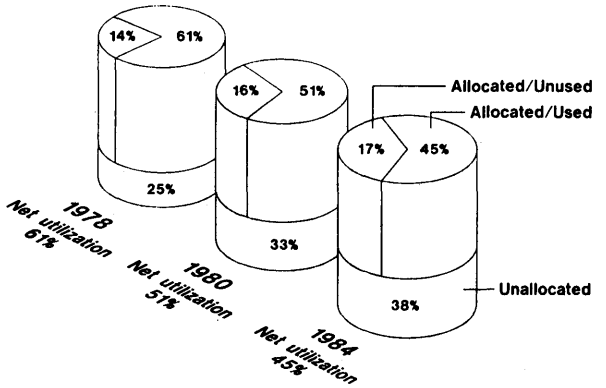


Figure 11. GUIDE DASD Usage Survey

Of the space allocated to DASD, not all is used. Over-allocation occurs if the data set is expected to expand requiring more space.and because of less than full-track blocking causing additional wasted space in count, key, data (CKD) format devices.

FUTURES

Future DASD directions include the implementation of smaller-diameter disks to reduce mass and provide a smaller form factor for making more effective use of valuable computer room space. The new, smaller form factors are likely to appear as a rack mounted device in the 1990 time frame. Smaller diameter disks also permit faster rotational speeds and therefore improved data transfer rates. Thin-film heads can be used to write narrower tracks and by using thin film or sputtered media, much higher aerial densities (megabits per square inch) are likely.
It is unlikely that the MVS operating system will support Fixed Block Architecture (FBA) in the near future. Other than VSAM, which comprises about 25 percent of the allocated DASD space (see Figure 12), there is little data that directly fits the FBA scheme. A feature of much promise is the actuator level buffer (ALB) which extends the concept of caching a step further to the actuator itself. The ALB virtually eliminates the latency issue by placing the data to be transferred in a high-speed RAM buffer. If the I/O path is busy at DASD reconnect time, the data is transferred to the ALB. As soon as the path is free, data transfer can begin without DASD latency occurring. Along with faster transfer rate capabilities comes the requirement for more concurrent I/O and increased availability. MVS/XA today supports up to eight paths to a device but no DASD subsystem yet exploits over two concurrent I/O operations. Future DASD will provide quad and possibly octal path connection capabilities.

Future DASD Directions

- *Smaller form factor*
- *Rack mountable*
- *Increased connectivity*
- *Higher availability*
- *Possible use of actuator level buffering*
- *Higher data porting capability*
- *Use thin film media*

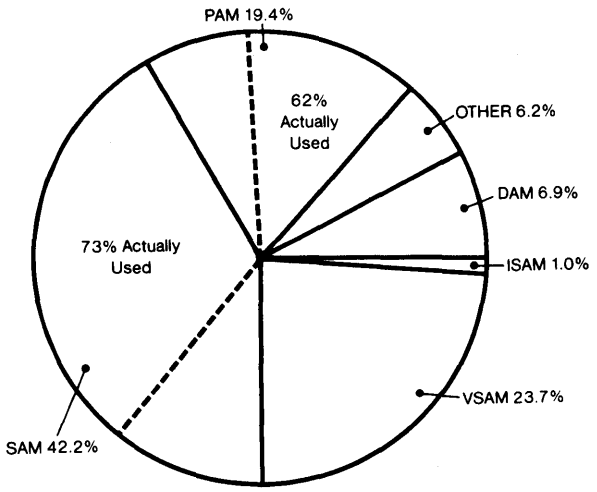


Figure 12.

MASS STORAGE

ARCHITECTURE (OVERVIEW)

Mass storage devices have existed for large computer users since the late 1960s with the IBM 2321 Data Cell. Since that initial offering, several mass storage products have been introduced though *none* have yet become a de facto standard for mass storage. Whereas high-performance DASD, cache DASD, and non-cache DASD all have access times measured in milliseconds, mass storage devices have access times measured in several seconds. Reliability and availability problems have surrounded mass storage products limiting their success. Suppliers of mass storage systems usually have chosen to discontinue such systems for a variety of marketing and technical reasons. Some mass storage systems have used a combination of accessors (picker arms) and data recording devices to transfer data from unique data storage cartridges or high-density video tape stored in a library. Others have chosen to use industry-standard media such as nine-track tape reels or, most recently, the 18-track cartridge. From an architectural point of view, using an industry standard media such as nine or 18-track tape has the advantage: If the mass storage system breaks down, the tapes can be read or written on any compatible nine- or 18-track drive. Today, the presence of robotics using software program control provides significantly higher reliability potential than some of the previous offerings. Improvements in recording density and packaging now provide the capability to connect several like mass storage systems together providing a nearly limitless storage capacity.

Mass Storage Systems should provide:

- *Relatively quick access*
- *Data access compatible with systems software and access methods*
- *Technologies which can be enhanced to provide long-term storage solutions*
- *System accessible storage media*
- *Cost effectiveness not only in price but operational and environmental measures*

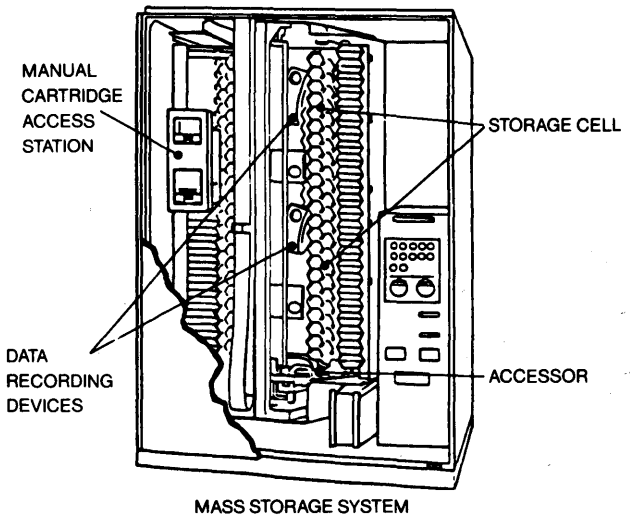


Figure 13. Mass Storage System

PERFORMANCE

The use of a mechanical accessor or robotic arm to access data provides performance improvements over manual mounting of media but cannot provide the sub-second access times of rotating DASD. Mass storage systems provide initial access times in the range of 10 to 25 seconds. The first question to consider regarding mass storage performance concerns the initial service time requirement. What is the time span between when the request for data is issued and the time the first byte of data arrives? If the answer is immediately, the data must reside on high-performance DASD (solid state) or DASD. If the answer is not immediately, the data can be stored offline. Although instant or online would be ideal, the decision-making process is influenced by the cost of online information and exacts a high price for a high level of initial service. The second question is what is the cost of storing the information? Data processing installations are faced with choosing online devices with a higher unit cost of storage, or offline devices with a much lower unit cost but a slow (sometimes several minutes) and inconsistent initial service time. Mass storage systems provide an excellent

solution for this requirement by combining the large capacity and low cost characteristics of offline storage with the quick initial access characteristics of online storage. The description of the performance requirements for the mass storage level are generally well understood. What remains to be seen is a truly successful implementation.

Mass Storage Performance Characteristics

- *Initial service 10-25 seconds*
- *Online or Nearline™*
- *Capable of bulk data transfer at current channel speeds*
- *Access by multiple requestors simultaneously*

Nearline™ is a trademark of Storage Technology Corporation

APPLICATIONS

The explosive growth of machine-readable information in the 1980s is likely to continue indefinitely. The advances in technology are opening new avenues for the use of information storage and retrieval throughout the business enterprise. The accumulation of historical information requiring long-term retention for legal and analytical purposes represents terabyte (trillion byte) storage requirements and is an area yet to be addressed effectively. Many organizations such as insurance companies must have access to all prior claims 10 years past or more. This amount of data cannot be stored effectively on microfilm or in file cabinets any more, due to its raw volume and labor-intensive management requirements. Technology has permitted the generation of information to grow. Today, point-of-sale terminals, automatic teller machines and electronic funds transfer provide for continuous information generation. A few of the more common applications ideally suited for mass storage systems include:

Generic Mass Storage Applications

- *Archive low-activity DASD files*
- *Scientific information*
- *Information required for long-term retention*
- *Record keeping*
- *Backup of select DASD files*
- *Numerous others . . . . . .*

FUTURES

Since a truly successful mass storage system does not yet exist, and the requirements for such a solution continue to mount, the future for such a system (if successful) is very promising. Technology improvements with program-controlled robotics, proven media, and microprocessors hint that a successful solution may be on the horizon. The most immediate solution would appear to be one employing proven storage media such as 18-track tape cartridges. This solution addresses the requirement for a storage media that can still be accessed should the mass storage system fail and removes human operators from the process of placing data on a channel. An automated cartridge system provides an ideal archive for low-activity DASD files, a major performance improvement over manual tape handling, and a cost-effective solution for new applications.

Another mass storage candidate is optical storage. A few false alarms and concern over the repeatability of re-reading optical media has caused concerns. Optical storage has the characteristic that it can be written only once and read many times in random or sequential mode. Re-writing data is not yet possible for optical, although this barrier for the large computer market should be passed successfully by 1990. Once optical possesses re-write capabilities, software applications will be able to access optical like rotating DASD with READ, WRITE and UPDATE functions at a lower cost per megabyte. However, due to positioning of the laser diode, the access times of optical drives can be five to 10 times greater than DASD. There are no flying height or head crash problems with optical and no contamination problems except contamination accidentally molded into the media itself during manufacturing. Should a reliable media be developed but not followed by the erasability, the future for optical storage is in application-specific markets and *not* for general-purpose use. Other storage media such as video tape exists though acceptance of such implementation has been slow.

The potential for mass storage systems is enormous. As the industry waits for a successful solution, the real success of a mass storage solution may very well rest in storage management software. The concern of how to place the right data on the right device remains unaddressed and requires a great deal of human involvement. The problems of today and tomorrow are identical to those of the past, but the sheer magnitude has increased by epic proportions. A storage management subsystem will be of utmost importance to the success of mass storage systems and to the management of the entire storage hierarchy.

Future Mass Storage Directions

- ► *Possess READ, WRITE and UPDATE capabilities*
- ► *Possess the ability to be connected for expanded capacity*
- ► *Low cost, automated access*
- ► *Access from multiple requestors*
- ► *No host operating system modifications*
- ► *Ease of data protection and recovery*
- ► *Provide a cost-effective storage management solution*
- ► *Use of fiber optic channel extenders for offsite archive*

MAGNETIC TAPE

ARCHITECTURE

Removable magnetic tape has been the media of choice for interchange backup and archive requirements for over 20 years. Magnetic tape storage offers high and continually improving storage capacity at ultra-low cost. Tape needs today consist of the traditional nine-track reel-to-reel tape and the newer 18-track cartridge tape subsystem. The 18-track cartridge system has a capacity of 200 megabytes and a nine-track reel consisting of 2,400 feet of tape can hold approximately 160 megabytes of data. The flux density has changed from 9,300 flux change per inch (fci) in nine-track to over 38,000 with the 18-track cartridge. It appears entirely feasible that track density could be doubled to 36 tracks per inch and both tape length and flux density increased to yield approximately a 1.0 gigabyte capacity for cartridge tape. The capacity of a nine-track reel is unlikely to change from the 160 megabyte level. Tape subsystems typically consist of two tape control units interfacing with four or eight host I/O channels and up to 16 tape transports or drives. Magnetic tape is erasable and overwriteable enhancing its usefulness and is a well proven and understood technology.

(See Figure 14)

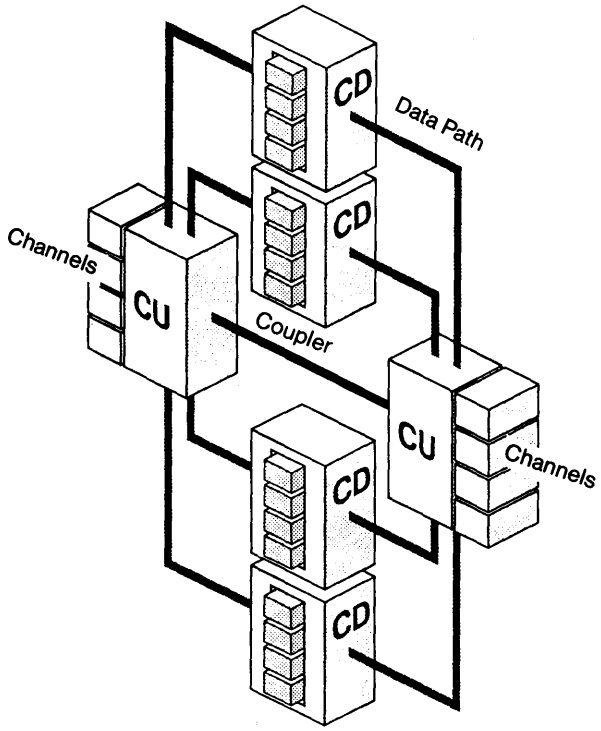


Figure 14.

PERFORMANCE

Access to manual tape is very slow compared to all other levels of the storage hierarchy resulting in initial access times ranging from one to three minutes. This is the mount time — the time it takes for a human to locate the reel or cartridge and mount it on an available tape transport. Eighteen-track systems can transfer data at today's channel speed limitation of three megabytes per second. These systems have the ability to transfer at higher channel speeds by using the tape cache to provide upward speed-match capabilities. Channel utilization for both 18- and nine-track tape subsystems can exceed 80 percent since tape does *not* suffer from the DASD RPS phenomenon of disconnect and reconnect. Though nine-track tape can only READ or WRITE data at a maximum of 1.25 megabytes per second, buffered tape control units allow transfer to occur between the buffer and I/O channels at three megabytes per second. The capability to buffer may have been the last major breakthrough for nine-track though the capability to buffer at faster channel speeds exists.

Tape Performance Summary

18-Track	9-Track
► *3 MB/sec data recording*	► *1.25 MB/sec data recording*
► *3 MB/sec channel transfer*	► *3 MB/sec channel transfer*
► *90 percent channel utilization*	► *80 percent + channel utilization*
► *20 ms start/stop time*	► *1.5 ms start/stop time*
► *Good with larger blocksizes*	► *Good with small and large blocksizes*
► *Caching at record level*	► *Buffering at record level*

APPLICATIONS

Tape is the primary data interchange media in use today. Ease of portability makes tape ideal for offsite storage and disaster recovery backup. The most popular use of tape today is for DASD backup. As information becomes more critical to the business, loss of information can be catastrophic; therefore, more backup occurs more often. Unlike DASD, tape has virtually no capacity limit and multi-volume tape data sets commonly exist. A survey of 20 large users conducted by Storage Technology Corporation in 1985 revealed the following distribution of tape data set sizes and uses. (See Figure 15)

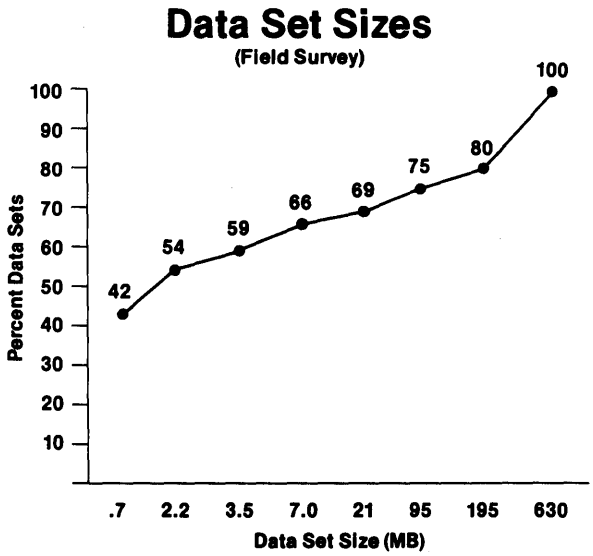


Figure 15. Data Set Sizes

Tape has also become the primary target for archiving low usage DASD files. In the absence of a successful mass storage solution, tape has had to assume the role of a partial alternative to mass storage. Few new applications are being developed for tape though the role of tape as *the* interchange and archive media is firmly established. (See Table 1)

Table 1. Tape Demand Profile Estimates (Field Survey)

Data Interchange	15%
Backup	25%
Archive	10%
Production	50%

— Few new applications written for tape

— More data is being backed up much more often

— Batch window down to a few hours for many users

FUTURES

It is likely that any future enhancements to nine-track tape will come primarily through supporting higher channel transfer rates using the buffer. Further advances in capacity for nine-track tape are much less likely to occur. By comparison, the future for 18-track includes a likely increase in track density to 36 tracks and further improvements in both tape length and flux changes per inch (linear density) resulting in higher capacities and transfer rates Continued advances in drive packaging are likely to improve the use of computer room floor space. Advances with fiber optic channel extenders will soon make it possible to extend tape connectivity to several kilometers away from the host computer. Coupled with automation, 18-track cartridge subsystems growth may prove to be a significant opportunity for cartridge tape.

CONCLUSION

The 1980 decade has brought a number of significant improvements to the storage hierarchy. From a user's perspective, these changes have brought improvements in price, performance and capacity while creating many new storage problems due to the vast amount of data to be managed. Clearly, technology permits this growth but does not drive it. The pressing issues for the remainder of the 1980s will become storage management. At present compounded annual growth rates, the magnitude of this problem — managing storage — may actually become a barrier to growth in the not too distant future. (See Table 2)

Table 2.

Growth Rate Projections				
	1984	1988	1992	CGR
MIPS	*28*	*112*	*448*	*35%*
GB (DASD)	*92*	*353*	*1279*	*40%*
Batch	*24*	*15*	*6*	*Percent*
Online	*55*	*62*	*69*	*workload*
Systems	*21*	*23*	*25*	*composition*

The challenge of managing over 1,000 gigabytes of online data will be aided to some degree by technology advances but the major responsibility will fall heavily into the area of software. A number of storage management software products exist that provide solutions to some of the problems but none offers a complete solution. In some user environments, several storage management products co-exist in attempts to balance weaknesses in one package with the strengths of another. For years the storage hierarchy has suggested hardware only. The storage hierarchy of tomorrow must include hardware and software to be ultimately successful. (See Figure 16)

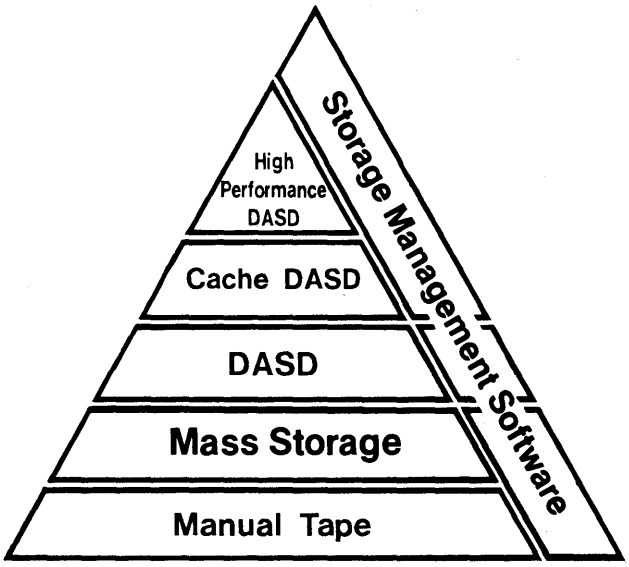


Figure 16.

Bull Moose Does TA:
A Manager's Guide to Transactional Analysis

Jim Stanton
Vice President
Morino Associates

ABSTRACT

Bull Moose is a highly successful technician recently promoted to the ranks of management. He has received no management training and has been learning the supervisory ropes by trial and sometimes humorous error. He has been the subject of a series of articles in the Morino Software News over the last few years and has fallen into most all of the managerial traps. Last year at CMG, the presentation on Bull Moose's transition into management received the award for best management tutorial. This year's presentation follows Bull Moose as he learns of a pragmatic technique for observing and dealing with both the natural and the unnatural behavior of others. He learns how to recognize dangerous political situations, identify and deal with "games" played by his staff and others, and learns how to deal more effectively with his peers and his boss. There is some of Bull Moose in us all, as you will see in this tutorial on using Transactional Analysis in the DP environment.

Introduction

First, we must introduce you to a character that all Information Systems (I/S) organizations can readily recognize -- Bull Moose. This is the technically brilliant, but temperamentally difficult individual who has saved the bacon for his superiors on countless occasions during their struggles to provide information technology to the enterprise.

The proud recipient of an enviable series of laurels for technical bravado, Bull wears the "MVS Availability During Prime Time" medallion with the triple oak leaf cluster. His left lapel sports the "TSO Response Time" badge with its twin silver stars.

His peers have unanimously named him to the "Putting Up With Management Honor Squad." And -- to the envy of systems programmers everywhere -- he is a recent inductee into the prestigious "Service Level Agreement Legion of Merit." It is widely rumored that Bull Moose knows more MVS record types than any technician anywhere -- from the highest mountaintop to the dankest basement -- wherever companies are held hostage by the computer industry.

In fact, the ability of our hero to single-handedly secure technical objectives while maintaining total disdain for management is the stuff of fairy tales. Bull Moose is, quite simply, a legend in his own time. That is, he was, right up until his recent promotion to Technical Services Manager.

A Deflating Promotion

In stark contrast to his former technical prowess, Bull has lately been spotted wandering all over the management battlefield, uncovering land mines that have blown his former self-confidence into oblivion. His medals have tarnished, the oak leaf clusters have wilted and drooped, and his silver stars have become millstones around his neck as he struggles to gain a foothold on the management ladder.

One by one, Bull's bold forays into the many delicate areas of technical management have ended with a sadder but wiser Bull. Thus far, we have seen Bull fumble his way through the snarly issues of corporate culture, department leadership, employee counseling, and most recently, staff recruiting. Like all intelligent managers, Bull is learning from his mistakes, but is coming to face the reality that as a manager, he will often have to deal with yet another brush fire before the previous one is completely out.

Bull Moose needs even more help, and fortunately this time some is on the way. As we catch up with Bull, he is about to reach a watershed in his "trial and error" approach to management. He is about to enter a new realm of enlightenment -- one in which he will discover one of the most useful tools ever developed for understanding and dealing with human behavior. Yet, every good lesson comes with some pain; thus, we are about to see Bull plop his right front hoof onto one of the biggest land mines in corporate America.

The Story

The atmosphere was calm in Bob's office as Bull arrived early for the usual Monday morning managers' meeting. Bob, Bull's boss, seemed cordial enough, but Bull felt uneasy. Several support problems had cropped up over the last few days and Bull was in a state of continual apprehension. He hadn't heard any encouraging words in weeks.

In fact, he had heard nothing but complaints as he and his group struggled diligently to support the Applications group in their attempts to implement a new corporate financial management system. The Applications group was in overdrive, running jobs around the clock, and generally driving Bull crazy.

Curt Camel, Manager of Applications Development, came in and took his usual seat. Bull thought Curt Camel was stupid. You could look him straight in the face and see that he was stupid. He seemed to be able to go on forever without a drink of knowledge; it was hard for Bull to contain his disdain for Curt and his plodding ways.

Sitting at the table, Bull could feel his antagonism rise as he watched Curt slowly

shuffle his papers, preparing for the meeting. He moved so slowly, so deliberately, as though he were in charge. Bull was getting angrier and angrier as he watched Curt.

Abruptly, Curt looked up, smiled, and said "How are you this morning, Bull?" Bull's anger peaked. Here was this Camel, who for the lack of ability to manage his stupid programmers, was causing Bull's group all kinds of grief over machine availability. And now, he was acting like he had Bull by the horns, smirking at him in front of the boss.

Bull rose to the occasion. "I'd be a lot better this morning, Mr. Camel, if someone were controlling the application development fanatics in their use of TSO. If they would spend more time checking their code instead of letting the computer stumble over their errors, we'd have more availability time for productive use." The smile slid from Curt's face to Bull's.

Curt responded: "Well Mr. Moose, I find it a shame that our company doesn't have the expertise to keep the gas tank full and the oil checked on our computer center. If it weren't faltering all the time, we wouldn't have nearly so many reruns." The sardonic smile crept back to Curt.

"We've got the best computer up-time in the industry," responded Bull as the smile, now waning, moved moved to Bull. "What industry?" countered Curt. "The funeral industry?" The smile had disappeared. So had Bob's patience.

Reading the Riot Act

"What's the matter with you two?" snapped Bob. "It seems to me that neither of you are meeting your schedules lately. If you'd spend less time bickering and more time solving problems, we'd all be better off!" The smile had moved to some other building. It would not surface again in this meeting.

Bob's diatribe gathered speed. Bull was devastated. Bob attacked him on several issues Bull didn't consider problems -- and surely not of a magnitude that concerned Bob.

One by one, others arrived for the staff meeting and found themselves submerged in the foul mood of the original three. No one was spared by Bob. Schedules were questioned, priorities challenged, the handling of issues Bull didn't consider problems -- and surely not of a magnitude that concerned Bob.

One by one, others arrived for the staff meeting and found themselves submerged in the foul mood of the original three. No one was spared by Bob. Schedules were questioned, priorities challenged, the handling of projects criticized. When anyone would try to raise a defense, Bob would swing deftly from his horse, lopping off heads with his sharpened saber.

After what seemed an eternity, the meeting ended, and Bull trudged back to his office, miserable. He couldn't believe some of the things Bob had said. And that stupid Curt Camel -- he had started it all. Bull was getting mad again. Then, Tom, the data center manager, came into Bull's office and closed the door behind him.

Misery Loves Company

"Man," he said, "what in the world got into Bob ? I saw him earlier this morning, and it seemed like he was in a good mood. We had a long discussion on the new procedures I want to implement for the remote sites. Then, in the meeting, he went after me on the same things we had just agreed on! What's wrong with that guy? His management style stinks. How does he expect me to do my job with a million changes in direction every nanosecond?"

"I know what you mean," Bull chimed. "Stupid Curt is his favorite manager, and now I think I know why. They're both stupid, and...."

This litany went on for some time. Soon, both Bull and Tom were starting to feel better. By the time Tom left, Bull was ruminating about certain of his staff members who were behind in their work.

Bull tramped over to Gail's cubicle and peeked over the top. "How are things going?" he asked. As Gail looked up, she thought: "Why doesn't he leave me alone? I already told him this morning before the meeting how things were going." She responded aloud to Bull: "Oh, about the same as I told you this morning. No changes in the last 10 minutes worth having a summit meeting about." The sarcasm in her voice was certainly not lost on the astute Bull Moose.

Nostrils flaring, Bull lowered his antlers and charged full speed into Gail. When he left, she was so angry and so fed up with Bull Moose and his constant heckling, his second-guessing, his directives to do it his way, his badgering style, that she couldn't see straight -- much less remember where she had left off on the intricate program problem she had been attempting to solve. She got up from her desk and made for John's cubicle, where she launched a quiet but stinging oratory on the probable nature of Bull's parents.

John looked up, surprised, then quickly entered into a supportive role with Gail. After all, this was a lot more interesting than the prolonged headache he was experiencing over his impossible-to-solve program problem. Besides, Bull hadn't done him any favors lately.

Just Another Day?

As we can see, a normal, potentially productive day has been disrupted to the point of no return -- and it's not even noon yet. What's happened to all our heroes? Has the world gone mad? Is this just another day in corporate America? Does this kind of nonsense really go on? You bet! Can any of this be prevented? You bet!

How? you ask. Well, several weeks ago, as

Bull was researching interviewing techniques at his local library, he stumbled across an introductory text on Transactional Analysis (TA). Intrigued, he took the book home, in hopes of gleaning some advice on how to work with the motley bunch of programmers in his group. As we witnessed in this morning's meeting, however, the principles and dynamics of TA have not begun to sink in on Mr. Moose. Let's see what would happen if Bull gained a working knowledge of TA through his readings, then began to apply it to the work environment.

What is TA?

TA is a tool for observing, understanding, and responding to human behavior. It provides a means for observing and analyzing verbal exchanges (transactions) between people. It is a positive communications tool that is practical and almost immediately useful. It can dramatically improve a person's on-the-job effectiveness by lending better self-understanding, greater insight into what makes other people tick, and a clear perception of what goes on when people try to communicate.

The beauty of TA is that it is not only easy to learn, but can also be applied successfully in personal and family situations. It can serve as a common language for people working together towards clearer understanding. It is a nonthreatening approach to self-evaluation that offers a method for analyzing not only people, but also organizational scripts.

The Ego States

TA is founded on the premise that everyone has three basic ego states: Parent, Child, and Adult. At any given moment (during waking hours), we are all in one of these states, dramatically coloring our interactions with others. How we communicate with others, how we receive their remarks, how we think, and how we react to the situations around us are all heavily dependent on our predominant ego state.

An easy way to understand ego states is by imagining three separate tape recorders in our heads. Each is continuously and simultaneously recording our feelings and the information presented to us by others -- called "stimuli" in TA terms.

"The Parent"

The first tape recorder represents the Parent ego state in all of us. The Parent provides us with our "taught" concepts of life. It contains recordings of imposed, unquestioned, external events we perceive from our birth up to age five. This data -- recorded straight and unedited -- is what gives us our "can-do" attitude. The Parent ego state, however, is fortified with feelings such as "never" and "always" -- typical words parents use with children.

The Parent provides us with some simple rules of living, and enables us to respond to certain situations without recalculating every time. For instance, our reactions to such dangers as a sharp knife or a loaded gun are part of our Parent ego. Also included are those parts of our personality that deal with values and opinions.

Within this ego state, we can easily identify two separate forms. The first is the Nurturing Parent. Here lie our yearnings to care for and protect, along with the unconditional acceptance of another's traits.

The second form is the Critical Parent -- that person who only accepts someone if he or she follows instructions closely. Certain verbal clues tell us when we are in the Critical Parent ego state. They include: "should, ought to, do, don't, and follow my example."

"The Child"

The Child ego state represents our "felt" concept of life. The Child consists of recordings of internal events, or feelings, that are a response to external events such as interactions with our parents. The Child contains our feelings of frustration, rejection, and abandonment. It also contains a vast storage of positive data. Certain verbal clues that tip us off when we are in the Child ego state include "I need, I feel, wow, gee, and I want it my way."

As with the Parent, there are two forms of the Child ego state. The first is the Natural Child -- that part of our personality that feels alternately crushed by disappointments and overjoyed by success. The Natural Child is the source of our feelings of self-doubt or inadequacy -- as well as pure emotions, creativity, and spontaneity. It provides the "want to" drive in all of us.

The second form of the Child ego state is the Adaptive Child, also known as the Compliant Child. Here is where we find thought patterns that can vent themselves as either rebellious or submissive behavior.

"The Adult"

The Adult ego state contains our "thought" concept of life. It's based on the gathering and processing of data acquired and computed through exploration and testing -- in, short, our life experiences so far. The Adult is a slow and continually developing state, and begins when we first begin to "think" about situations -- at about 10 months of age, rather than at birth.

The Adult comprises the rational part of our nature. Rather than relying on outdated parental dictums, the Adult in us acts as a computer, digesting current factual data for problem-solving purposes. The Adult often plays the role of executive, utilizing both Parent and Child data for decision-making and authorizing the other ego states to become dominant when appropriate.

The Adult knows no right or wrong; rather, it deals strictly in fact. It evaluates situations by looking at how decisions are derived -- not at the decisions themselves.

It examines the Parent and Child data and performs probability estimating in an attempt to maintain consistency in words and actions. It updates and validates the Parent, Adult, and Child data bases.

Conflict, when it occurs, causes a great deal of "data processing" in the Adult; unpleasant alternatives, in particular, cause considerable computer lag. Verbal clues to being in the Adult ego state include: "who, what, how, why, and let's consider this."

Thus, the three ego states give us our perceptions of the world. The Parent provides the rules for behavior from both the Critical and Nurturing frames of reference. The Child provides intuition, creativity, spontaneity, and enjoyment. And finally, the Adult provides the "data processing" and evaluation of options, in addition to regulating and coordinating the Parent and Child in us.

Here's the catch. We are constantly moving from one state to another, sometimes in response to what others say to us, or by how they say it; other times the shift is a result of our own perception of what is going on. When we change from one ego state to another, we say we have been "hooked," into the new predominant state. It is the dynamics of "hooking" and "being hooked" that make transactions so interesting.

Transactions

Whenever we speak to someone -- whether verbally or with gestures -- we provide a stimulus. This, in turn, engenders a reaction, or response. These activities -- stimulus and response -- are referred to as "transactions." Of these, there are two basic types: complementary and crossed, as illustrated in the accompanying diagram.

In Figure 1, you can see that complementary transactions result when the stimulus and response are parallel. In Figure 2, crossed transactions are occurring -- the result of a stimulus and response that are not parallel.

When we objectively consider Curt's question, "How are you this morning?," we can see that it was an Adult-originated transaction, targeted for Bull's Adult ego state. Curt was simply being polite. Had Bull responded with "I'm fine, Carl," we would have had complementary transactions between two Adult ego states. This exchange could be diagramed as in Figure 1.

Instead, Bull's response to Curt came from his Natural Child ego state and was aimed at Curt's Nurturing Parent ego state -- counter to the initial transaction. When such crossed transactions occur, real communication ceases. While the transactions themselves may not stop, the transfer of meaningful information has ground to a halt. So, while Bull and Carl pursued their verbal sparring, they had, in reality, ceased to communicate.

"Strokes"

Transactions -- parallel or crossed -- serve another purpose aside from the simple exchanging of information. They can also be used to convey "strokes." These are the feelings we transmit to others through our communications. Just as a cat purrs contentedly when stroked gently with the hand -- as opposed to aggressively with the vacuum cleaner -- people experience feelings of contentment when they receive positive communications about themselves. In the context of TA, however, strokes can also be negative and impart bad feelings to recipients of a message.

In fact, there are four basic kinds of strokes:

1) Positive - "You did a good job."

2) Negative - "You did a lousy job."

3) Plastic - "You did the best job I have ever seen in my entire life!"

4) Discounted - "You did a good job, for a change."

When we get strokes, we "file them away," in TA terms, for later reference. A good analogy is that of a stamp collector. For every genuine positive stroke we receive, we get a gold stamp for our book. If we are angered, we slap a red stamp into our book. Depression resulting from negative strokes leaves us with a gray stamp; while feelings of self-righteousness and indignation earn us white stamps.

Everybody needs strokes and everybody wants to collect gold stamps.. Those who don't get sufficient strokes from others are forced to take them by other means. One way they do this is by cashing in some stamps, that is, trading them with others for strokes. This is where it gets interesting.

Hooking Others' Ego States

In the case of severe "stroke deficiency," people will often enter different ego states in an attempt to "hook" or force others into whichever ego state will result in their collecting strokes. Let's see how this worked in Bull's infamous managers' meeting calamity.

Let's go back to the moment when Bull entered Bob's office. We recall he was feeling uneasy -- a lack of positive strokes. In fact, Bull was consumed by self-doubt and stress, stemming from his recent difficulties with the Applications group. Bull perceived -- correctly or not -- that the incidents reflected badly on his group and his leadership capabilities.

Without his knowing it, Bull's ego state had shifted from Adult into Natural Child, seeking comfort, approval, and unconditional acceptance. The Natural Child signals Bull put forth were intended to hook into someone's Nurturing Parent, where protectiveness and comforting strokes abound.

Instead, Bull's Child response to Curt's

attempted Adult transaction hooked straight into Curt's Critical Parent -- not at all what Bull had intended. The resulting admonition from Curt led straight into Bull's Rebellious Child outburst, and they were off to the races.

What Was Bob's Role in This Mess?

Bob had been doing just fine, rolling along in his Adult ego state until the skirmish between Bull and Carl broke out and hooked Bob's Critical Parent ego state. Bob's subsequent invective hooked the Child ego states of all those in the meeting. Each participant was put on the defensive, collecting not the positive strokes they had anticipated, but a barrage of stamps -- red, gray, and white -- which were stashed in their stampbooks for later use.

Now let's look at the stamp collecting score of the various participants. During his Parent-to-Child blitzkrieg, Bob collected several white stamps for his feelings of self-righteousness and a dozen or more red ones for his anger. The unfortunate subjects of Bob's wrath, caught in their Child egos, collected a fistful of red stamps for anger and an additional sprinkling of gray for their despair and anxiety.

The Stamp Collections Grow

When Bull and Tom collaborated in Bull's office after the meeting, they continued collecting stamps. At that point, they were moving from their Child into their Parent ego states and entered readily into a favorite TA game called "Ain't It Awful." After collecting many red stamps with Bob's name on them, they gradually moved on to garner a few white stamps of self-righteousness for the moral indignities suffered at Bob's hands. In the process, they exchanged positive strokes by agreeing that Bob was wrong, and they were right.

Bull, feeling considerably better after his session with Tom, then took his bulging stampbook into Gail's cubicle, where he cashed in several red and a few white stamps at her expense. In so doing, he hooked Gail out of her usual Adult state and into her Rebellious Child, where she took a licking and collected a hailstorm of red, white, and gray stamps.

When Bull finally left, her anger had reached such a level that she stormed from her office to John's, where she was quickly able to hook John out of his Adult and into his Nurturing Parent. There, she succeeded in trading several red stamps for a few gold stamps of agreement, which helped to smooth her ruffled feathers.

As we leave the scene, John is headed for the water cooler, where he will try to unload some of his red stamps on someone else for a few positive strokes....

An Ounce of Prevention?

The irony of this whole disquieting scenario is that it could have been prevented. Any one of the players in the game could have simply refused to leave his or her Adult ego state, and focused on getting everyone else back into theirs. Or, when appropriate, he or she could move intentionally into whichever ego state would best deal with a problem transaction until the other party was able to return to an Adult state. Since the Adult state is where most productive business transactions and professional communication should take place, a good management meeting could have been ensured.

Bull can see this now, as he finishes Chapter One of his TA manual and reflects back on the dynamics of the meeting. Unfortunately, Bull still lacks sufficient insight into TA to fully understand why he feels the way he does, when he does or how to change his ego states at will. He isn't equipped just yet to put the brakes on a fracas like the managers' meeting. But, he has a good start!

James R. Bowerman
Technical Manager
Morino Associates Inc.
Vienna, Va.

Abstract

Historically, one of the most difficult problems encountered by the capacity planner has been the forecasting of the resource requirements of specific applications. A popular approach has been to use business element data (sometimes called natural forecast units or key volume indicators) to develop relationships between the volume of business activity in the sector which uses the application and the amount of resource consumption. This paper will explore the techniques which have evolved for developing a model to relate business activity to computer resource usage, and suggest how to initiate a business element forecasting plan in an organization.

Introduction

Capacity planning has been defined as the process of configuring a data processing system to meet the organization's business requirements. This is a very broad definition which has been interpreted to mean a great many different things by practitioners and theorists over the lifetime of the discipline. Generally the process of capacity planning is subdivided into four basic areas, understanding the current usage of the system, understanding the future requirements for the system, determining the ability of various configurations to support specified levels of usage, and relating the cost of these configurations to the value of the services provided.

Over the past few years a number of capacity planning methodologies have been developed. The literature abounds with articles promoting the use of various techniques, and studies which analyze the success and failure of these techniques in various environments. To date, a single comprehensive methodology has not emerged which can be used to implement a capacity planning program which is widely accepted by all practitioners.

It is the author's opinion that the lack of a comprehensive methodology is due to the very broad nature of the problems addressed by the capacity planner. The capacity planner must deal with technical issues such as relative CPU power, memory sizes, and I/O configuration. In addition she or he must understand the business requirements which drive the demand for the hardware and software resources. The planner needs a background in financial concepts so that cost comparisons may be made. Finally the planner must understand the structure of the applications which run on the system so that business requirements can be translated into hardware and software configurations.

Business element forecasting is one technique in the arsenal of tools which can be used to attack capacity planning problems in the I/S management organization. The purpose of business element forecasting is to relate business requirements to the demand for resources of specific applications.

The term "Business Element" refers to a data element which measures the amount of activity in a business division which uses I/S resources. Some authors use the term "Key Volume Indicator" or KVI to refer to this measurement. Other authors use the term "Natural Forecast Unit" or NFU. The differences in terminology can be confusing, but the underlying concepts are the same in most published approaches.

Business element forecasting is implemented by building statistical models which relate the amount of activity in a business division with the computer resources required to support the activity. Once the model is built, the capacity planner can forecast future values of the business elements, and employ the model to determine the level of capacity needed to satisfy these requirements.

The motivation behind this technique is the need for the capacity planner to look outside the I/S organization for estimates of future capacity requirements, and to quantitatively tie future I/S expenditures to the business plans of the divisions which I/S serves. If this process can be successfully implemented it provides the quantitative information needed to make management decisions about future I/S expenditures, since the cost of I/S resources can be directly related to the value of the business plans in the user divisions.

A business element forecasting approach is not the only method which can be used to plan for capacity requirements, nor is it an appropriate method for use in all situations. This method can be used in cases where an organization in the corporation uses a substantial amount of computer resources in the course of their production operations, and where the volume of activity in that organization is related to the use of these resources.

For example, an automated order entry system, where the sales force enters customer orders directly into a data base system which is then used to feed the manufacturing, warehousing, shipping, and billing processes, is a good candidate for business element forecasting. A system which is used by wide variety of corporate divisions, for a number of purposes, such as general purpose TSO system, may not be driven by business requirements in other areas. This type of system may be better forecasted by studying historical usage trends.

Requirements for the Methodology

The successful implementation of a business element based forecasting program will require the I/S organization to have already established some basic parts of a capacity planning program. The information which is needed is a definition and characterization of the workloads on the system, and an understanding of the divisions within the corporate organization which are served by these workloads.

The development of this characterization of the workloads is part of the natural evolution of a capacity planning program. Howe (HOWE85) gives a more detailed discussion of many of these issues and outlines how to approach the initial development of a capacity planning program. To better understand the base upon which a business element forecasting program is built, consider the following evolution of a capacity planning program.

Analysis of Global Utilization

The key ingredients to any capacity planning program are a basic understanding of what levels of resource usage are being attained, who is using the resources, and what future growth is expected in the usage of these resources. This understanding forms the basis for the capacity plan which addresses the relationship of business requirements to a final data processing configuration.

When an I/S organization initially implements a capacity planning function the first objective is to start collecting data on usage of the computer system so that this information can be tracked over time, and trend analysis may be performed. Tools currently exist for organizing data from system logs and monitors, such as SMF and RMF, so that this data may be extracted and analyzed.

In the early stages of a capacity planning program the I/S organization may only collect statistics on the global usage of system resources. Trends will be made on total CPU utilization, aggregate volume of online DASD storage, number of logged on TSO users, total volume of batch jobs, or total volume of online DB/DC transactions. These statistics show the aggregate usage of all I/S resources, and hence are related to the total costs of the installed hardware and software.

The best use of aggregate statistics is to analyze the profile of system usage so that peak periods may be identified. In today's online and competitive business environment, there is need to plan the maximum capacity of the system around the demand for resources in peak usage windows. While aggregate statistics are useful for identifying peak usage periods they are notoriously weak for trend analysis. Trends based on aggregate usage can show a need to increase capacity, but they cannot relate the need for additional capacity to specific business requirements. This means that aggregate statistics do not carry sufficient information upon which to base management decisions on upgrade expenditures.

Understanding Technical Subsystems

The next phase of a capacity planning program is to break down the aggregate statistics into their component parts. The first cut at this effort is often based on technical, rather than business, concerns. The data sources used for this analysis are specific to the technical implementation of the applications. For example a CICS online application is tracked by CMF, a batch application is tracked by SMF, a program development effort by RMF or TSO/MON, and the like. This activity allows the planner to understand the different data sources, and to learn what quantities are measurable within the applications.

Trends based on a technical breakdown of the resource usage add more illumination to the capacity planning process. These highlight growth by technical subcomponent, and may suggest alternate methods for providing new capacity. For example, if program development is showing significant growth, one may choose to migrate developers from TSO to VM/CMS, since CMS capacity can be acquired in smaller increments than additional TSO capacity (using 93xx or 43xx scale processors).

This analysis provides more technical detail and begins to suggest how solutions to capacity requirements may be engineered. Analysis at this level may sometimes address a business requirement, if the business requirement is the driving force behind a particular technical subcomponent of the resource usage. For many applications a more specific relationship to the business plan is still required.

Add Usage by Business Segment

Once the technical subsystems are understood the capacity planner can move into a planning environment which relates resource usage to specific business divisions. This involves breaking down the resource usage of the technical subsystems into components which are used by the various corporate divisions which use the I/S services. The goal of this analysis is to determine who is using the resources, and to quantify their level of consumption.

At this level of granularity the planner begins to have the information needed to present the business case for a capacity upgrade. A specific capacity requirement can be stated in both business and technical terms. The business usage is stated by major application, or by division within the organization. The technical usage is stated in terms of major subsystem usage, with individual statistics on TSO, batch, DB/DC, etc. The information can be visualized as filling in the cells of the following resource usage or demand matrix:

Columns: Technical Subcomponent
Rows: Business Division consuming the resource
Cells: Resources consumed (CPU seconds, EXCPS, transactions, etc.)

	TSO	Batch	CICS	IMS	.....
Finance	x	x			
Engineering	x	x	x		
Manufacturing		x		x	
Sales	x	x	x		
Marketing	x				
Etc.					
.					
.					
.					

Obviously, the corporate divisions, and the technical subsystems will be chosen according to the structure of the corporation, and the technical requirements for computer resources. Usually more than one matrix of this type will be required. Each matrix will contain data for a specific time period, or time zone, upon which the capacity plan is based. These matrices will be used in various parts of the capacity plan, and provide input to

guide the business element forecasting process. The primary matrices which are needed are:

1. Load during peak periods
2. Load during prime shift
3. Total load for all shifts

In an online environment the total capacity of the system must be configured to support the peak demand periods, which typically occur in the mid-morning and mid-afternoon. This means that the costs born by the I/S organization are determined by the demand for resources during peak periods. The second two matrices are needed for correlating the business element data and for supplimentary management reporting.

Targeting the Business Element Forecasting Effort

The analysis of global system utilization leads to identification of the peak periods. Further analysis of the load by technical subsystem or application adds insight into the specific subsystems, usually the online systems, which provide the demand during these peaks. When the demand matrix is completed the planner knows which divisions of the corporation are responsible for this demand.

The business divisions which use significant portions of available resources during the peak periods are the ones which are targeted for further investigation. These business divisions are the ones which drive the requirements for additional capacity. As such, quantification of the benefit derived from this usage is the key to providing the justification for capacity upgrades.

A demand matrix for the prime shift is needed for correlation with business elements which are obtained outside the I/S organization. This is because the capacity planner must obtain historical data on values of the business elements from the business divisions which use the resources.

Unless business divisions outside of I/S maintain very detailed statistics on their volume of activity by hour of the day, or by week of the month, it will not be possible to obtain business elements which correspond exactly to the peak periods of resource demand. Since most activity occurs during standard business hours it is much easier to correlate business elements with prime shift loads.

In some environments, especially manufacturing, where production occurs around the clock, it may be necessary to correlate business activity with total I/S resource usage without regard to time. These environments need to construct a demand matrix for total load over all shifts.

Business Element Methodology

The basic hypothesis which underlies the business element forecasting methodology is that the volume of business activity in the corporate divisions which are served by the I/S organization may be related to the demand for computer resources. Furthermore, the methodology assumes not only that this relationship exists, but also that it may be quantified using a statistical model so that estimates of the volume of future business activity may be directly translated into specific quantities of computer resources.

The methodology can be subdivided into three parts. The first part identifies the business elements which can be obtained from the users of the I/S resources. The second part uses statistical procedures to relate, or correlate, historical observations on business data with computer resource usage. This analysis leads to the development of a model in which computer resources are represented as a function of the values of the business elements. The last part of the methodology involves forecasting future values for the business elements. These forecasts can used as input to the model developed in the second part to predict computer usage.

Identifying Business Elements

Successful identification of the business elements is, perhaps, the most difficult part of the methodology. This is because the capacity planner must work with the users of the system to determine the elements which the users feel best characterize their levels of activity. The capacity planner must shed her or his perceptions of the computer system and learn to see the application from the user's perspective.

This is where the peak period demand matrix is used. The capacity planner cannot hope to interact with every business division which uses I/S resources. By studying the peak period demand matrix the business divisions which drive the capacity requirements can be quickly identified and the process of identifying business elements can be focused on these divisions.

The next step is for the capacity planner to interview the developers of the application to determine the characteristics which they believe have the most significant influence on the resource requirements of the system. This type of information will include the number and type of transactions which are executed to perform a particular business function. The capacity planner is interested in developing a conceptual overview of the logic of the system which relates end user function to measurable resource quantities.

Understanding the logic of the application will not yield the business elements, but rather give the capacity planner the perspective to discuss use of the system in the terms most familiar to the end user.

Once the system is understood, the users of the system should be interviewed to determine how they measure their volume of usage of the system. This interview should be viewed as an iterative process. This interview is intended to lead to identification of the business elements themselves. To successfully implement this methodology the capacity planner needs not only to identify possible business elements but to obtain historical observations on these elements for correlation to the historical patterns of resource usage.

The capacity planner should recognize that planning resources to meet business requirements is an activity which is practiced in all parts of the corporation. In the I/S arena, we plan hardware upgrades, raised floor space, power and cooling capacity, staff levels for system programmers, and similar things. The business divisions which use the I/S services also plan to meet their production targets in units such as office space, staff levels, number of machines, capacity of production lines, and the like.

As a starting point, ask the users what statistics they report to their management to quantify the level of work processed by their department. When the business division is requesting additional staffing, or capital investment, what units do they use to justify these needs? What statistics are used to regularly track their progress against targets established in their business plans?

In some industries, most notably the insurance industry, the volume of future business activity can be very accurately determined. The financial management of an insurance company is absolutely dependent on the ability to forecast future claim activity. As a result, insurance companies have actuarial departments devoted to this activity. The capacity planner should seek this type of information whenever possible.

The advantage to using statistics which business divisions use for tracking and reporting on their own level of activity is that historical observations on these statistics are readily available in the user's management reporting. This provides the data needed to build the business element model. In addition, these statistics are the same measures, in the same units, that are used to formulate the business plans of the business division under study. A model based on these statistics ties the user's business plan to the capacity plan for the I/S organization in a very natural manner.

Preparing to Build the Business Element Model

The process of building a model to relate the business elements to the demand for computer resources has two sources of input. The first source are the historical observations of the business elements which were obtained in the previous step of the methodology. The second source are historical observations of computer resource usage.

The data needed is the same data used to build the demand matrices which break down resource consumption by business division and technical subsystem. This data is available from system logs and monitors such as SMF and RMF for MVS systems, CMF for CICS, the IMS log tape, and other similar measurement products.

Historical resource statistics should be extracted over the same time period from which historical business element data was collected. These statistics should be summarized to the same level of granularity as the business element data. If the business element data shows monthly aggregate activity volume, then the resource data should be summarized to monthly level. This process creates a common time scale which is critical to the use of the statistical correlation process.

Formulation of a Business Element Model

A business element forecasting model is a linear equation of the form:

$$Y = A0 + A1*X1 + A2*X2 + A3*X3 + A4*X4 +...+ An*Xn$$

In this equation the variable Y represents the observed value of a resource data element, such as CPU time or total EXCP's. This is called the dependent variable. The variables X1, X2, through Xn represent the observed values of the business elements. These are known as the independent variables. The A0, A1, through An are coefficients which relate the business elements to the resource element.

This model can be interpreted to mean that resource consumption, Y, can be expressed as some underlying constant, A0, plus a factor, A1, times the value of the first business element, X1, a second factor, A2, time the value of the second business element, X2, and so on. This model is expressed in a form which can be solved using multiple linear regression, a common statistical technique which is implemented in most statistical packages. The goal of the multiple regression analysis is to estimate the values of the coefficients, A0, A1, through An, to completely specify the model.

The choice of a multiple regression model is motivated by several factors. Foremost is the simplicity of the model and its intuitive appeal. In addition, multiple regression models are widely used in various disciplines to model this relationships of this type. The form of the model is similar to models used in medical research to relate risk factors to incidence of disease. Regression models are also widely used in economic analysis and in psychological research.

The form of the model shown above assumes a purely theoretical relationship, where the resource consumption is exactly related to the values of the business elements. Since our data is collected from a real world environment we cannot hope to obtain an exact relationship. The model which we will attempt to estimate can be written in the following form:

$$y = a0 + a1 * x1 + a2 * x2 + ... an * xn + u$$

This is the same form above as above, with the dependent variable Y, replaced by the observed values of historical resource consumption elements denoted by lower case y. In a similar manner the theoretical values, X, of the business elements are replaced by the sample of data gathered from the users, which are denoted by a lower case x1, x2, through xn. The estimated values of the coefficients are denoted by lower case a1, a2, through an. Finally we add an error term, denoted by u, which contains the residual difference between the linear combination of the coefficients times the observed values of the business elements.

Basic Assumptions

Several basic assumptions are needed for the statistical formulation of this model. These are:

1. the x1, x2, through xn, are statistically independent
2. the theoretical mean value of the error term is zero
3. the error term has a constant variance
4. the error terms are statistically independent
5. the error term is independent of the x1 through xn

These assumptions introduce some statistical terminology which requires definition. A complete definition of these terms is beyond the scope of this paper, but can be found in most texts on linear regression (KEL74). To develop a business element model a full theoretical understanding of these terms is not required. To judge the validity of the model the capacity planner does need to understand the importance of these assumptions.

Assumption 1 requires that the business elements be statistically independent. This implies that each business element represents a unique data point which cannot be expressed as a function of one or more of the other business elements. The statistical term for a model which fails to meet this assumption is multicollinearity.

Failure to satisfy this assumption implies that you have not found a complete set of business elements upon which the model may be estimated. When the business elements are related to each other the model will not be able to determine which of several related business elements have predictive power on the dependent resource element.

Independence of the business elements is a major problem in formulating a business element model. You should intuitively expect a high degree of relationship between the business elements. For example suppose you have collected business element data on sales contact activity, number of orders entered, number of units manufactured, and the number of customer shipments.

These are all very naturally related to each other. The independence of these elements is highly suspect, and will depend on the size of the inventory maintained by the corporation, and the length of the sales to shipment cycle. If the inventory is low and the sales cycle is short there may be a very strong correlation between all the elements. When inventory is high, or if the cycle is long, then the correlation will be weaker, in that order entry is correlated to sales activity in an earlier month, manufacturing appears to be more driven by the need to maintain inventory, and the like.

Assumption 2 requires that the error term have a theoretical mean of zero. Failure to meet this assumption implies that there are other factors, aside from sampling and data error, which influence resource consumption. This means that the group of business elements which have been identified are incomplete. The full predictive power of the model cannot be realized until the unknown factors are identified.

Assumptions 3 through 5 all concern the statistical properties of the error term. Taken together they require that the error term be a well behaved random variable which can be quantified using statistical techniques.

A variance is a statistical parameter which controls the amount of dispersion of a variable about its mean value. To say that the error term has a constant variance is to say that the size of the error term is bounded in probability. This does not imply that there is a maximum value to the potential error, but rather than the probability of seeing an error greater than a specified value is small. In statistical terms, the model is said to be homoscedastic if this assumption is met and heteroscedastic otherwise.

Like multicollinearity, heteroscedasticity is a common problem in relating business element data to resource usage. As an example consider the problem of tracking corporate divisions of different sizes. A very large division typically has a large budget, more discretionary capital, and a strong motivation to exploit economies of scale. These factors suggest high variability from mean computer resource usage. An extreme example is an automotive assembly line which runs at maximum capacity for the first half of the model year to build inventory, then shuts down entirely for retooling for the next model year. A smaller division does not have this flexibility, so you would expect lower variance in their resource usage.

The final assumptions concern the independence of the error terms from other error terms and from the values of the business elements. Like the other assumptions this assumption assures us that the model contains all the factors which influence on the dependent variable. This assumption is violated when there is a systematic pattern to the error terms in the model. This problem is known as autocorrelation.

Autocorrelation is not typically a major problem in business element models. Where autocorrelation does cause problems is in using regression models to track historical trends over time. To illustrate consider a tracking of a division which operates under very tight budget constraints. If the resources usage in any month exceeds their budget allocation they restrict usage in the following month to recover the costs. This may put them behind schedule, thus causing them to exceed the budget in the third month. The error term in this model will swing from positive to negative and back in each successive month.

Solution Techniques

With the form of the model specified there are two questions which need to be investigated by the statistical procedure:

1. Which business elements should be used in the model?
2. How can reliable coefficients be estimated?

The first question arises from the process used to collect the business element data. In interviewing the users the capacity planner has sought business element data which appears to be related to resource usage. The choice of elements relies heavily on the judgement of the user and ability of the capacity planner to understand what drives the resource usage of the application. The choice of business elements is also influenced by the need for historical data. In many cases the capacity planner will be using historical data based on its availability more than its intuitive appeal for forecasting.

The issue of which business elements to include in the model can be addressed through the use of stepwise regression techniques. A stepwise regression is used to survey a large number of possible choices for independent variables and select the subset which creates a model which best fits the historical trend of resource consumption.

There are a number of different stepwise techniques which can be used. All of these techniques select a subset of the independent variables, solve the model, and calculate a statistic which measures the strength of the model. The value of this statistic is then used to weigh this subset of independent variables against other subsets. If the current subset is the best possible subset the process stops.

The technique which is used in the SAS system (SASS85) is known as the MAXR approach. This approach starts with a pool of business elements and evaluates every business element individually to determine which yields the strongest model. The selection is based on the value of r-square, with the highest r-square being chosen as the best model.

The r-square is formed by calculating two measures of variability. The first is the sum of the squared differences between the predicted values of the dependent variable from the mean value. The second is the sum of the squared differences between the observed values of the dependent variable from the mean. The ratio of these values may be interpreted as the proportion of the variability of the dependent variable which is explained by the model.

The technique proceeds to evaluate all two element models, all three element models, all four element models, and so on, until it determines that the value of r-square cannot be increased by adding more business elements to the model.

The result of this procedure is a model based only on the business elements which have the strongest predictive power. The procedure also estimates the coefficients which are applied to business elements, and generates a number of statistical tests of the strength of the model.

Validating the Model

The strength of the business element model may be determined in several ways. For a business element model to be useful it must have good predictive power on the dependent variable. This means that the model must explain a large percentage of the variation in the dependent variable.

Beyond predictive power in general, the business element model must be capable of predicting changes in the dependent variable as each business element variable changes. This means that the models needs to be tested in each of its multiple dimensions to determine the validity of each coefficient on each business element.

Lastly, the model should provide good estimates of future values of the dependent variable. This is related to the overall strength of the model in which we stated that the model should explain a large percentage of the variation. This variation effects the level of confidence which the capacity planner can have in the predicted values. Intuitively, being able to predict a future value within 20% is useful, while predicting within 400% is totally worthless.

Overall Strength of the Model

The strength of the model as a whole is judged on the basis of the level of the r-square which is attained, evaluation of the F-statistic generated by the statistical package, and the judgement of the capacity planner.

As discussed earlier the r-square measures the amount of variability in the dependent element which is explained by the model. If all the variability is explained, then the model is perfect, in that there is no error in prediction. The r-square test is not, however, an absolute test. A model can have poor predictive power but still generate a high value of r-square.

The F-statistic is a random variable which measures the probability that the model would have arisen due only to chance. To phrase this another way, make the basic assumption that there is no underlying relationship between the business elements and calculate the probability of seeing the results which have been produced. If the probability of seeing this F-statistic under this assumption is very small, usually less than 0.05, then you can safely conclude that there is a valid relationship expressed by the model.

The final, and often the most important, test of a model is the judgement of the capacity planner. The planner must ask herself or himself if the results of the model make sense. When your intuition and experience are in line with the model you are on the right track. If this is not the case, then either the model is wrong, or your intuition is lacking. The situation requires further investigation.

Testing Individual Business Elements

When the overall model has been validated, the capacity planner should turn her or his attention to the coefficient estimates for the individual business elements. Ask yourself whether there is sufficient justification for each business element to be included in the model.

The t-statistic from the statistical package is used to perform this test. This is another random variable which measures the strength of each coefficient estimate, in the same way that the F-statistic measures the power of the entire model. To apply the test, make the assumption that the coefficient for a particular business element is zero. This is to say that business element should really not be used in the model. If the probability of seeing generated value of the t-statistic is small, or less than 0.05, you can reasonably conclude that this business element has predictive power.

One of the most common problems in business element forecasting is a high level of correlation between the supposedly independent business elements. This violates the first of the basic assumptions of the model, and leads to a situation known as multicollinearity.

The classical tracks of a multicollinearity problem are very good statistics on the overall model, combined with low t-statistics on all or most of the business elements. In this type model the r-square and F-statistics are almost always highly significant, so the problem is not detected until you attempt to validate the coefficients on the individual business elements.

In a multicollinear model the statistics are trying to tell you that when you consider all of the business elements together, as a group, that they have a great deal of predictive power. However, this predictive power is so confounded by relationships between the business elements that the solution techniques cannot determine the predictive power of any single business element. This makes the model useless for forecasting, since you cannot expect to add future estimates of the business elements and get reliable results.

Testing the Predictive Power of the Model

The final test of the validity of a business element model is in its ability to predict future levels of resource consumption within a reasonable tolerance. This involves calculation of what are known as confidence limits for the predicted values.

A statistical model cannot actually predict the future, it can only estimate the most likely, or expected value for a forecast. This is a probabilistic estimate of the average, or mean, value of a random variable. Confidence limits are a boundary which is put around this predicted value to establish the probability that the observed value in a future time will fall within a certain range.

The important criteria for a confidence limit is the width of the limit around the predicted value. Generally confidence limits are drawn with either a 90% or a 95% probability.

To illustrate the use of a confidence limit assume that a model has been built to predict the CPU time required by an application, and that the statistical package has generated an estimate of 12.5 hours/month, with a 90% confidence limit of plus or minus 2.3 hours. On the basis of this model, the capacity planner can make the statement that there is a 90% probability that the future value of the CPU time is between 10.2 and 14.8 hours/month.

This model is clearly more useful than it would have been if the confidence limit was 10.8 hours. Had the confidence limit been 10.8 hours, the capacity planner would have only been able to say that the future value of CPU time would fall between 1.7 and 23.3 hours. This interval is so wide that the estimate is of little value.

Generating Predictions

The final phase of a business element forecasting study is the use of the model to predict future resource requirements. The model takes as input the predicted values for future levels of business elements. It then produces estimates, and confidence limits, for the level or resource consumption which is to be expected. Therefore the input needed for the forecasting study is predicted values of the business elements.

There are two ways to obtain estimates for future values of the business elements. One approach is to go back to the business divisions who provided the initial historical data upon which the business element model was based. These divisions should be forecasting their own requirements as part of their normal planning progress. If you are using the same statistics that the business divisions use for tracking their activity the forecasts will already be available.

This approach will provide the strongest business case for your capacity plan. If the plans which the business division have already been approved by corporate management, your capacity plans have all the justification they need, as they are additional requirements which are directly in support of a committed business plan. If the plans for the business division have not been approved by corporate management, your capacity plans should be presented in conjunction with the plans for the business division. This gives management the information that they need to make a quantitative decision.

The second approach to forecasting business elements is to apply any available forecasting technique directly to the historical values of the business elements. A large number of possible techniques are available in commercial software packages (SASE82, MAKR83). Several published papers have explored this technique. Hoffman (HOF82) showed an application of a three component model which measured trend, season, and cycle to the message rates of an airline reservation system. This technique showed that a simple model could be used to predict future volumes upon which a business element model could be based. Lo (LO86) explored the use of different types of forecasting algorithms which use data smoothing on historical series of business elements.

In either case, be aware that the accuracy of a business element forecasting study depends on both the characteristics of the statistical model proposed, and on the accuracy of the forecasts provided by the business divisions which measure the business units. Each of these items of must be periodically reviewed.

The process of obtaining estimates of future business elements can be proceduralized as part of the regular corporate planning cycle. This will help ensure that the activity receives the attention that it requires in the user's business division. Salesmanship is important in obtaining cooperation from the user. The goal of a successful business element forecasting study is the accurate planning of computer resources which is highly beneficial to the user.

The statistical model can be rederived periodically from updated observations on the business elements. Since it is a statistical model, the exact values of the coefficient estimates will change slightly each time the model is run. This should not be a concern if the amount of change is small. Should the coefficients change greatly the basic model should be redeveloped using the same procedures used to develop the original model.

Conclusions

This paper has been a introduction to a business element based capacity planning methodology. This methodology requires the I/S organization to have a capacity planning project in place, which has identified peaks load periods, understands the technical subsystems, and can separate measurement data from various business divisions which use the I/S resource. The evolution of a capacity planning project up to the required point has been traced briefly.

The basic assumptions of a business element forecasting methodology have been stated, and a procedure by which the methodology may be implemented have been described. This procedure is composed of three major parts, identification of business elements, development of a model, and generating predictions. Suggestions have been given for identifying and collecting business element data, for using statistical tools, and for how the model can be used to generate forecasts.

References

ART84 Artis, H.P., The Relationship Between Installation Accounting and Performance Management and Capacity Planning, 1984 CMG Conference Proceedings, Computer Measurement Group, Alexandria, VA, 1984.

ART81 Artis, H.P., A Data Management Algorithm for MVS Capacity Planning, 1981 CMG Conference Proceedings, Computer Measurement Group, Alexandria, VA, 1981.

ART79 Artis, H.P., Forecasting Computer Requirements: An Analyst's Delimma, EDP Performance Review, Applied Computer Research, August 1979.

DUN74 Dunn, O.J., and Clark, V.A., Applied Statistics: Analysis of Variance and Regression, John Wiley and Sons, New York, NY, 1974

HOF82 Hoffman, L.L., Workload Forecasting Using Econometric Time Series Analysis, 1982 CMG Conference Proceedings, Computer Measurement Group, Alexandria VA, 1982.

HOWE85 Howe, D.J., A Basic Approach to a Capacity Planning Methodology, EDP Performance Review, Applied Computer Research, Phoenix, AZ, March/April 1985

KLE72 Kelejian, H.H. and Oates, W.E., Introdution to Econometrics, Principles and Applications, Harper and Row, New York, NY, 1974

LO86 Lo, T.L. and Elias, J.P., Workload Forecasting Using NFU: A Capacity Planner's Perspective, 1986 CMG Conference Proceedings, Computer Measurement Group, Alexandria VA, 1986

MARK83 Makrudus and Wheelwright, Interactive Forecasting, John Wiley and Sons, 1983 (User's Guide for the SYBIL/RUNNER system.)

SAR79 Sarna, D.E., Improving Computer Forecasting Using Key Volume Indicators, AFIP Conference Proceedings, Vol 48, 1979.

SASS85 SAS Institute Inc., SAS(R) User's Guide: Statistics, Version 5 Edition, Cary, NC: SAS Institute Inc., 1985.

SASE82 SAS Institute Inc., SAS/ETS User's Guide, 1982 Edition. Cary, NC: SAS Institute Inc., 1982.

THE USE OF NATURAL FORECASTING UNITS

John M. Reyland
Community Mutual Blue Cross & Blue Shield
Cincinnati, Ohio

ABSTRACT

The use of Natural Forecasting Units (NFU's) within capacity planning programs has proven to be an effective procedure. This tutorial considers the identification and measurements of NFU's and their use to project future computer system requirements based on planned or statistically projected NFU changes. It continues with a general report of a real experience utilizing NFU's, how their use not only projected future computer system resources, but also reported unexpected capacity and performance problems for a converting IMS system. The tutorial is structured around two different approaches in the use of NFU's and the capacity planning program.

INTRODUCTION

The NFU is a business element measurement from the business side of an organization which can be related to the use of computer systems resources by workloads. Examples of potential NFU's are purchase orders, production output counts, insurance contracts, accounting data, etc.. The purpose of NFU's can be twofold. The first deals with measuring and evaluating current system workloads as to their capacities and performance relative to selected business elements. The second deals with the impact of planned NFU changes as to their future impact on system configurations, workload capacities and performance, capital investments and budgets.

T. L. Lo [1] succinctly defines the NFU concept as attempts to link business activities/plans to DP activities/plans. Such linkage enables the data processing function to develop DP plans based on corporate plans, which is very important because of large capital investments needed to maintain adequate computer support.

The number of manufactured widgets might be directly related to a particular system workload and its CPU time. This would be expressed as a CPU time unit per manufactured widget. If this is a valid functional relationship between these two measurements, then one would observe during a given time period that a change in the number of manufactured widgets would cause a corresponding change in the total CPU time for the particular workload. Such a coorelation makes it possible to predict future system resources by workloads.

It is essential that one understands the functional relationship between NFU's and computer system resources by workloads. This is the basis of the NFU concept and its use to predict future computer system requirements. The above example of the academic widgets is simplistic, but it illustrates this functional relationship. The system resource side of this relationship is sometimes referred to as Data Processing Units (DPU's). Thus the relationship is between NFU's and DPU's.

THE NFU COMPONENT

Depending upon the business environment, there are many possible approaches to measuring and evaluating NFU's, which will functionally relate to computer system resources by workloads. There are some general considerations when selecting potential NFU's.

The first consideration is measurability, which is the ability to identify and count the occurrences of NFU's. This must be done with ease of identification and with consistency from reliable sources.

For purposes of this tutorial, NFU's can be categorized as contractual services or product lines. An example of a business industry that provides contractual services would be the insurance industry. Product line business would be the manufacturing of cars or television sets.

Identification deals with the terminology about contractual services and/or product lines. This can prove to be difficult, especially in the areas of contractual services. Services like medical insurance plans, which have evolved over time, have an ever changing realm of terminology. Terminology difficulties can also be found in product line businesses. The history of product diversification, model numbers, product specifications can all contribute to the problem of identification. Such instability makes it difficult to identify and track NFU's over long periods of time.

Consistency deals with the measurability of NFU's across business segments of the same organization. This is especially true within organizations that provide essentially the same service by its business segments, but the service has been tailored for particular client groups. This is probably a lesser problem in product line businesses.

The second consideration deals with the quality of the functional relationship between a potential NFU and its DPU. This quality is statistically referred to as correlation. Examples of the quality of correlation are illustrated by the following environments:

* The best measure of correlation can be found in a dedicated online system, which has individually identified transaction codes that support specific services.
* Correlation would diminish if the above services were supported by an online system that simultaneously processed multiple functions with a limited number of transaction codes.
* Batch processing would probably provide the least amount of correlation. To identify and count types of input records by type process is a less dependable procedure.
* If contractual services and/or product lines do not have direct data processing support, then correlation must be determined by some indirect means. This might involve sales records, accounting data or the use of manufacturing resources.

The ability to identify and use NFU's from the business side of an organization is greatly dependent upon one's understanding of the business organization. One should study the formal, and if possible, the informal structure of the business organization. Within this study, one must be careful to identify line and staff functions. It is within the line function that most NFU's will be identified, even if measured by a staff function such as accounting. The ability to identify and measure NFU's will also depend to great extent on the dynamics of an organization. Business and non-business organizations must continuously change to be competitive.

It is risky to select NFU's by a trial-and-error approach. The risk involves the reliability of selected NFU's, and the credibility of the capacity program. NFU's only become reliable when they have a history of being useful and dependent upon to make valid projections of required computer system resources.

COMPUTER SYSTEM RESOURCE COMPONENT (DPU)

The identification of a computer system resources unit (DPU) is found by examining workloads that directly support an NFU. This identification can become difficult because workloads may contain composites of work. A batch workload may contain both test and production work. An IMS system may have multiple IMS systems, or a transaction may perform multiple functions and call other transactions. To avoid this kind of problem, it is essential to identify specific workloads by assigning them to unique performance groups within the IPS of the System Resource Manager, a component of the IBM's MVS operating system. Non-IBM computer systems need to have similar features to isolate workloads.

Performance groups can be used to isolate types of workloads, which also isolates system resource utilization by such workloads. Depending upon the type of workload, not all system resources are reported by SMF/RMF at a great enough detail to be useful.

SMF/RMF will report by performance groups system resources utilization for batch workloads. Typically this includes CPU times, execution times, elapsed times, allocation times and for devices their I/O counts and service times, etc.. There still remains hundreds of other batch workload measurements.

TSO workloads are very much like batch. In addition to typical system resources being reported, TSO also reports Puts and Gets as terminal activity, and in some environments, TSO commands.

Beyond batch and TSO, data about the utilization of system resources must come from other dedicated sources. IMS and CICS have their own programs to reduce log data to meaningful reports. IMS PARS provides extensive data about individual transaction codes, elapsed times, processor times, data base calls, etc.. IMS also has DB/DC real time monitors that provide even greater detail than data reduction programs for log files. CICS systems have similar capabilities to obtain detail data about transactions and system resources.

There are communications packages that sample, and in some cases, report actual system resource utilization which can be used for communication workloads.

Most IBM compatible software vendors do provide some SMF recording about system resource utilization and transaction counts. This data, along with standard SMF and RMF data, can provide enough data to include such workloads within a capacity program.

NFU PROJECTIONS

There are several approaches in the use of NFU's to project the need for future system resources.

The first approach is to relate NFU's to input counts such as IMS or CICS transactions. These counts are used to drive a queueing or simulation model to change workload demands within a model. Projected IMS or CICS transaction counts are determined by an NFU conversion factor, which is calculated by relating NFU's and actual transaction counts. Future system resource requirements are determined from the changes within the model.

Such models usually require a preprocessor program to reduce SMF, RMF or other system data to workload characteristics. These programs will also generate input files for models, which contain required system resource measurements by workload. Morino's model generator and BGS's Capture program are examples of such software.

The second approach is to statistically project NFU's for purposes of determining changes in DPU's. Such changes are then translated into new hardware and software requirements.

NFU CONVERSION FOR MODELING

The needed functional relationship for the NFU is between the NFU counts and the model's input transactions. Such a relationship could be between policy holders and actual IMS transactions. There is also a need to mark this relationship at a point in time, possibly month end. All counts must also be scaled down as required by modeling software, which usually operate at hourly intervals. The selected hourly interval can represent a peak processing period or can average value.

More importantly the relationship must be functional. If there is an increase in policy holders and such policies are supported by a dedicated IMS system, then an increase in policy holders will cause a corresponding change in the number of IMS transactions. Such IMS transactions can be categorized as those that update policy information, inquire against the policy holder's databases and process policy holder's claims. The opposite is true, a decrease in the number of policy holders would result in a decrease in the number of supporting IMS transactions by transaction category.

These relationships must correlate either directly or indirectly. They may be simple, involving one NFU component or a string of individually weighted components. The more complicated a function becomes, the more difficult it is to measure and validate the relationship. The complexity of such relationships would be determined by the business environment. The mathematics usually involves the division of the transaction counts by the NFU component. This produces a conversion factor of IMS transactions per policy holder during the same time interval. This relationship can be expressed by categories of supporting IMS transactions. Conversion factors could be developed for IMS transactions per inquiry, IMS transactions per claim and IMS transactions per miscellaneous task. For each conversion factor there must have been an identifiable system workload with dedicated performance groups and IMS transaction codes.

Conversion factors from one time period to another should have approximate values. Historically they should have statistical distributions, which can be evaluated by standard statistical methods.

An example of a conversion process will serve to bring together the concepts about NFU's and their use for modeling. The example relates policy holders to IMS transactions. The process counts the NFU's, policy holders at month end and then scales selected IMS transactions to hourly averages. The NFU's are categorized as claims processing and non-claims processing. Dividing the hourly IMS transactions by the respective NFU will result in two conversion factors. The first would be average hourly IMS transaction for claims. The second would be the average hourly IMS transactions for non-claims processing. This assumes that the IMS system has been set up so that IMS transactions used to process claims have been uniquely defined and assigned to message processing regions (MPR's) and performance groups. The same would be true for the non-claims processing.

Sequence of steps to complete the conversions and utilize the NFU for projections:

* Assign IMS transaction codes to MPR's as claims and non-claims processing. Assign MPR's to dedicated performance groups.
* Count IMS transactions by selected IMS transaction codes needed for claims and non-claims processing.
* Scale IMS transactions to hourly rates. This can be done by having available hourly counts that correspond to the desired hour selected to be used by the model. If hourly IMS transaction counts are unavailable, then daily IMS transaction counts can be scaled by use of measured factors. Experience in our environment indicates that 12.78% of the daily IMS transactions will provide a peak hourly rate of IMS transactions.
* Perform the respective divisions to obtain the conversion factors for the hourly number of IMS transactions per claim and non-claim processing.
* Define the performance groups to the enqueueing or simulation model and execute the data reduction programs needed to process SMF and RMF data for the model.
* Distribute IMS system resources to IMS workloads within the model. This is necessary because SMF and RMF recording of IMS I/O's and CPU times are aggregate values for a processing period. By means of the DC monitor, the percentage distribution of these system resources can be determined. Such IMS resources usually involve the CPU times of the control region to MPR's and the distribution of I/O's by DDNAME to storage devices.
* Validate the model from actual system performance values and system measurements.
* Project the IMS workloads by using the conversion factors with planned changes of NFU's. This involves the multiplication of the monthly conversion factor by the planned monthly changes in NFU's. This is done on a monthly increment, since NFU's and IMS transactions are counted at month end.
* It is assumed that the model is being executed with all its other workloads, and that they are changing or remaining the same during the projections. It is also assumed that with the current workloads, the system is operating at over 85% CPU utilization and that there is a shortage of DASD storage.
* As the NFU's are converted to greater number of IMS transactions, the system would saturate with infinite response times because of the lack of system resources. It would be necessary to add system resources and to reconfigure the total system to process the increasing number of IMS transactions, while maintaining desired performance levels.
* The model's report of changed system resources are the future system requirements to process the planned NFU changes.

A REAL EXPERIENCE WITH NFU's AND MODELING

The above example is representative of our NFU program during the past 18 months. The purpose of our NFU program is to track, analyze and make recommendations for future system requirements, as part of a multi-year conversion effort. The conversion involved the replacement of an older IMS health insurance system, which does not have online claims processing, to a newer system with online claims processing and other desirable features.

There have been three stages which can be reported on in general terms, as means to present some realism of an on-going NFU program.

The first stage was early in the conversion schedule. Actual sample sizes of converted contracts were nonexistent. In place of these samples, there was another health insurance system, similar to the new system, which was used to obtain system utilization and IMS data. There was not an official consolidated schedule for the conversion of contracts by the five business segments of the organization. By means of a telephone we were able to consolidate the individual conversion schedules.

The contract counts functionally related to IMS data. The conversion factors were stable and useful to compute IMS transaction levels as input to the model. The results of the model indicated an inordinate amount of system resources needed to process online claims in the outer time periods. This was disconcerning, but our measurements lack sufficient sample sizes and we used a mirror image of the new system for system utilization and IMS data.

The second stage proved to be very much like the first, because very little of the scheduled conversion had occurred.

The third stage was the start of a new game. A group was established to officially set a conversion schedule and track its progress. In doing so, a new term appeared that was different than ours. The term contract was replaced with the term policy holder. Technically they are different, which changed our initial NFU measurements. At this point we had to align our NFU measurements with the new terminology, and recalculate conversion factors.

With better sample sizes, an official schedule of NFU's, good functional relationships and conversion factors, we were able to do more reliable modeling. The output of the model still indicated an inordinate amount of system resources needed to process online claims.

This became the center of concern for the business segments, data operations and systems and programming. Capacity planning and performance management suggested a limited amount of online claims processing with the remainder in batch. This was unacceptable to the business segments because they had been promised full online claims processing.

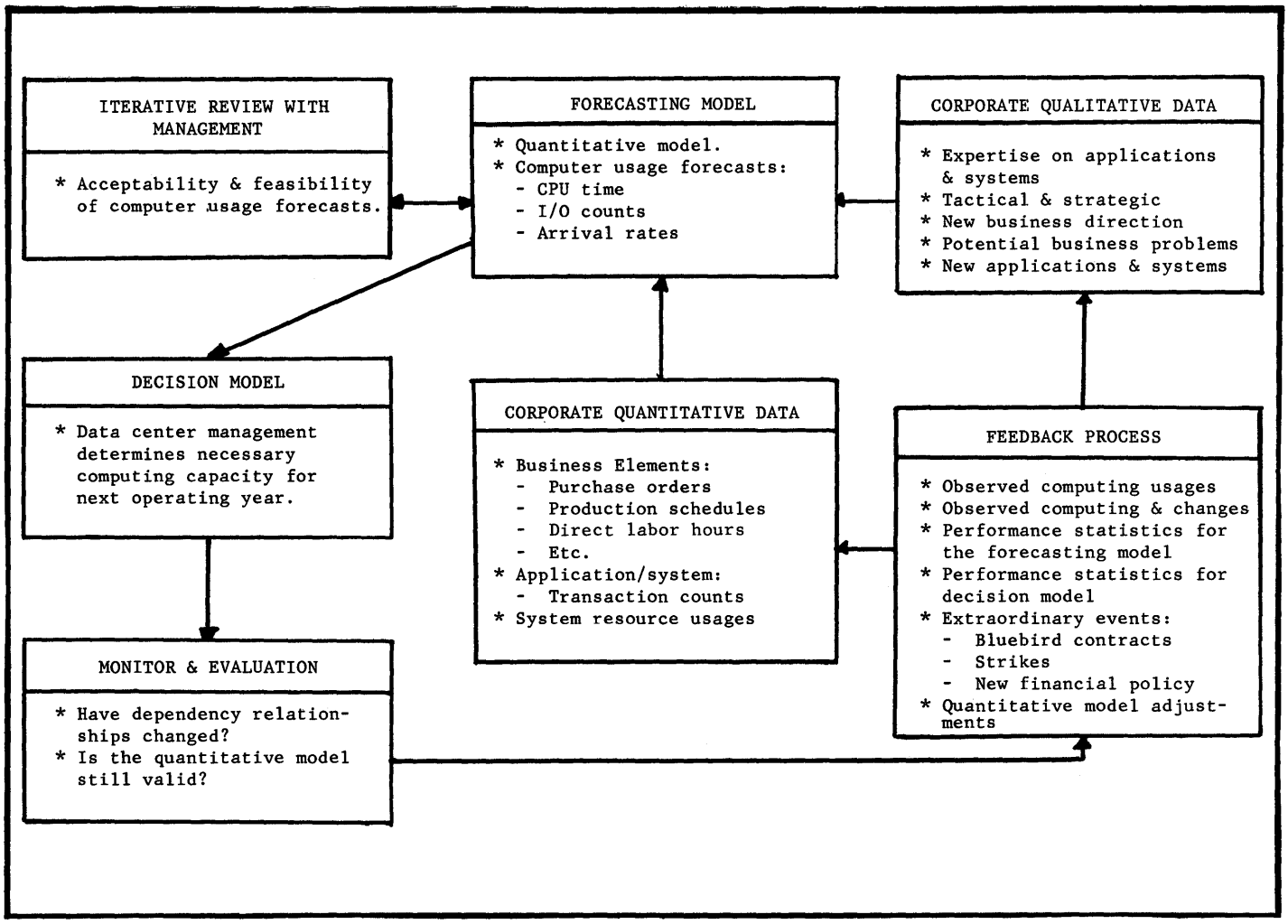
The system designers, data base people and programmers have studied and corrected this performance by a code change for claims processing. Today our projection models are reporting a totally convert system on one side of a 3090/400E with good response times.

NFU FOR DPU STATISTICAL PROJECTIONS

T. L. Lo [1] made a presentation at the CMG86 conference which gave details about their capacity program and how they statistically projected NFU's to determine DPU's. In summary, their capacity program has the following steps:

* Identify business elements at NFU's and data collection.
* Determine DPU's and their data collection.
* NFU/DPU dependency analysis.
* Forecasting process.
* Determine capacity requirements.
* Iterative review and revision.

The following chart, Lo [1] is a high level flow of their capacity program.


ITERATIVE REVIEW WITH MANAGEMENT
* Acceptability & feasibility of computer usage forecasts.
FORECASTING MODEL
* Quantitative model.
* Computer usage forecasts:
- CPU time
- I/O counts
- Arrival rates
CORPORATE QUALITATIVE DATA
* Expertise on applications & systems
* Tactical & strategic
* New business direction
* Potential business problems
* New applications & systems
DECISION MODEL
* Data center management determines necessary computing capacity for next operating year.
CORPORATE QUANTITATIVE DATA
* Business Elements:
- Purchase orders
- Production schedules
- Direct labor hours
- Etc.
* Application/system:
- Transaction counts
* System resource usages
FEEDBACK PROCESS
* Observed computing usages
* Observed computing & changes
* Performance statistics for the forecasting model
* Performance statistics for decision model
* Extraordinary events:
- Bluebird contracts
- Strikes
- New financial policy
* Quantitative model adjustments
MONITOR & EVALUATION
* Have dependency relationships changed?
* Is the quantitative model still valid?

The center piece of their program is the forecasting model and how it interfaces with other managerial functions of the flow. The forecasting model is designed around the dependency of the NFU and DPU, which is usually a linear relationship. In general, the forecasting process involves the following steps:

* Collect the NFU data from a historical source and use smoothing techniques to reduce randomness of the input data while determining projected values.
* Pass the smoothed NFU projected values to a Boole & Babbage program, the Workload Planner. This program produces projected DPU values (CPU times and I/O counts) from a historical source of system data. Such values are generated by a stepwise regression that uses the most affecting NFU's.
* Projected DPU values are used to determine future hardware and software system requirements.
* This process is iterated through the management flow for monitoring and evaluation, and for feedback.

CONTRASTING THE NFU APPROACHES

The first approach, preparing NFU's for modeling, is practical in smaller business organizations, which are stable and have a limited number of potential NFU's. This approach is particularly good in those system environments where the performance and capacity planners have control over the SRM parameters.

The second approach, projecting NFU's with statistical processes, is used within large business organizations having many segments that dynamically change their organizational structures and strategies. Such organizations potentially have a large number of NFU's that probably do not directly relate to unique workloads.

REFERENCES

[1] T. L. Lo and J. P. Alias, "Workload Forecasting Using NFU: A Capacity Planner's Perspective". 1986 CMG Conference, Dec. 1986.

AFTER THE STUDY IS OVER

Craig A. Cooley
National Advanced Systems

ABSTRACT

Today's complex data processing environments have driven most companies to implement capacity planning functions. However, only the largest corporations can afford to have a full-time staff dedicated exclusively to performance reviews and workload capacity projections. As a result of this corporate posture, the systems programmer dons the hats of the capacity planner and the performance analyst. Frequently, the systems programmer will provide overly detailed technically oriented reports that are meaningless to the upper level decision-making management within a corporation.

This paper will provide some examples of reporting techniques that will prove effective for the technician who has to make recommendations to non-technical decision makers.

1.0 INTRODUCTION

Mark Twain said "Get your facts first, then distort 'em as you please."[1] The preceding quotation is a summary of what can happen in the computer capacity planning environment when technicians are attempting to communicate with non-technical managerial decision makers. The technician becomes extremely frustrated and begins to tell the non-technical person what he thinks that manager wants to hear rather than what is needed to make effective corporate decisions.

When data is presented to non-technical management in simple to understand formats the non-technical person will feel as though he has a full understanding of the material. This will also make the non-technical manager feel that he is making the decision to purchase new equipment. When the non-technical management feels that this decision is theirs, the performance analyst and capacity planner have done their job effectively.

The intent of the remainder of this paper is to provide the technician with some simple techniques that will bring about the required configuration changes through mutually agreed upon understandings of simply prepared data.

2.0 FUNCTION

The organization diagram in FIGURE-I depicts a typical MIS reporting structure employed by many companies. Generally anyone from the MIS director up in this type of organization has been recruited from non-technical ranks and may have only a cursory knowledge of computing systems.

MIS ORGANIZATION

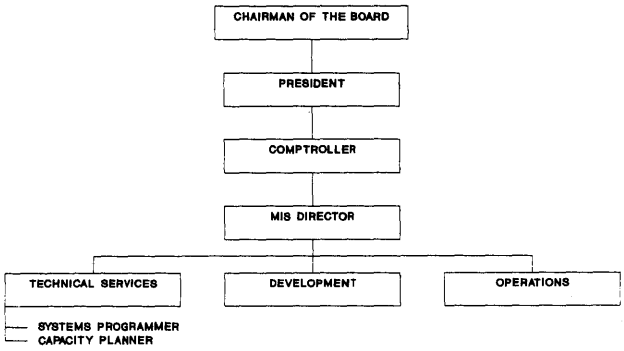


FIGURE-1

The capacity planner and performance analyst's functions found in the lower levels of the organization chart require them to review detailed data that is reported in some of the following technical terms:

- **SERVICE UNITS**
- **SIO'S**
- **MAIN STORAGE OCCUPANCY**
- **PAGING RATES**
- **PAGE MIGRATION RATES**
- **MEGABYTES**
- **GIGABYTES**
- **BYTES OF DATA TRANSFERRED**
- **BYTES OF MAIN MEMORY CONSUMED**
- **% DEVICE BUSY**
- **QUEUE's**

These terms are significant when talking to one's peers; however, when discussing these items with non-technical management, the performance analyst or capacity planner must show these items in units that non-technicians understand. Some of these units are as follows:

- **DOLLARS**
- **HOURS**
- **CHARACTERS**
- **GROWTH POTENTIAL**
- **TRANSACTIONS**
- **LETTERS**
- **INSURANCE POLICIES**
- **INQUIRIES**
- **ELEPHANTS**
- **ITEMS SOLD**
- **PATIENTS SERVICED**

As a result of these differences in terms, the general capacity and performance reporting procedure in these organizations becomes an interactive process as outlined in FIGURE-1A. This interactive process requires additional time from all parties involved in the reporting process. When the time spent passing through this process can be minimized, the corporation can gain significant hardware and "peopleware" productivity. These savings range from not having to hire an additional intermediate level manager as an interpreter, to obtaining six months, or better, of corporate cash flow through the timely acquisition of computing center resources. The following case studies introduced in this paper are the author's attempt at showing ways the technician can reduce the reporting cycle.

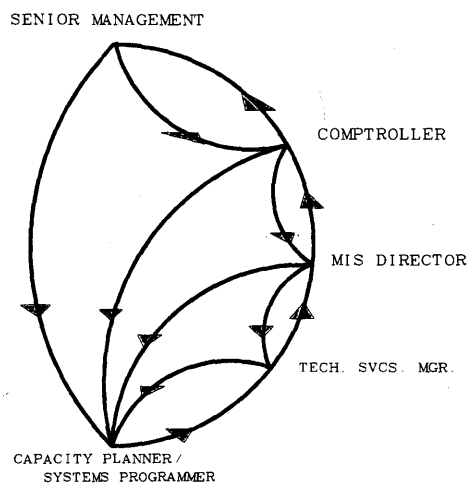


FIGURE-1A

3.0 REPORTING PROCEDURE

3.1 CASE STUDY 1

FIGURE-2 is an excerpt from a management report prepared by a competent MVS systems programmer. The intent of this report was to provide MIS management with processing figures that could be used as CPU capacity planning input during the annual budget process. Please note the similarity of the data presentation to the RMF monitor reporting format.

CPU UTILIZATION

	AVG. CPU % BUSY				CICS SERVICE UNIT			
	A:10	B:10	C: 2	D:B+C	WORKLOAD (MILLION)			
DAY	DATE	TO 4	TO12	TO 4 /2	IOC	CPU	MSO	TOTAL
FR	12/05	00.0	00.0	00.0 00.0	0.00	0.00	00.00	.00.00
TH	12/04	81.9	85.7	87.5 86.6	12.73	13.85	111.15	139.37
WE	12/03	78.3	84.9	73.8 74.3	13.36	14.82	118.50	148.40
TU	12/02	70.1	76.5	82.2 74.3	13.69	14.55	116.80	146.86
MO	12/01	69.9	81.0	69.0 75.0	14.28	14.36	115.84	146.32
FR	11/28	42.0	51.2	34.4 42.8	8.96	10.24	82.90	103.34
TH	11/27	HOLIDAY THANKSGIVING DAY						
WE	11/26	63.3	96.7	57.9 77.3	8.95	10.21	81.41	101.90
TU	11/25	62.0	79.5	70.7 75.1	13.94	14.24	114.56	144.62
MO	11/24	56.1	54.6	69.4 62.0	14.33	14.87	121.08	152.24

FIGURE-2

The MIS management team who received this report was completely bewildered by what the technician was trying to tell them and was reluctant to introduce any system changes, such as a verbally recommended CPU upgrade, until they understood the data.

The systems programmer's report contained less than one year's worth of actual data. These figures, plus some additional sketchy historical statistics, were then entered into a spread sheet program and reduced to the visual report shown in FIGURE-3. This visual report was then submitted to the same management team.

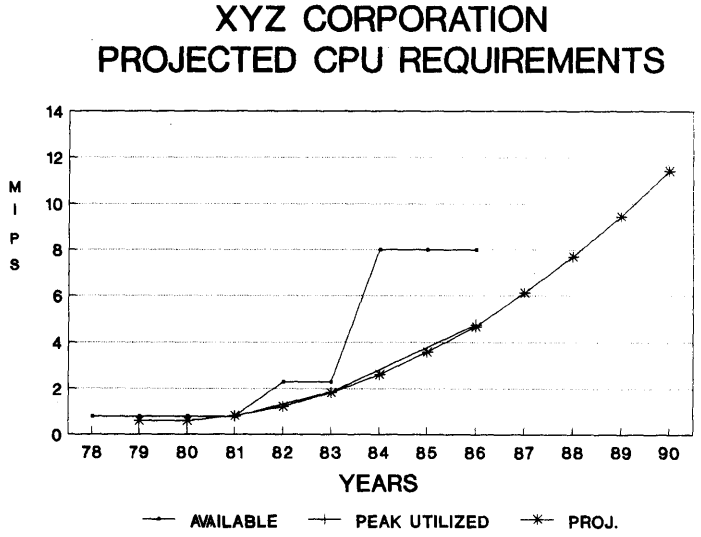


FIGURE-3

The graph presented in FIGURE-3 prompted the management team to reveal some new projects under consideration. These projects created CPU resource requirements in excess of the normal growth. The charts were then updated to reflect the additional growth items and the graph in FIGURE-4 was produced. Note the identification of a growth range in this report.

With FIGURE-4 available, the management team was able to start budgeting for CPU upgrades. This final capacity projection visual was included in the document that was submitted to the budget committee for approval.

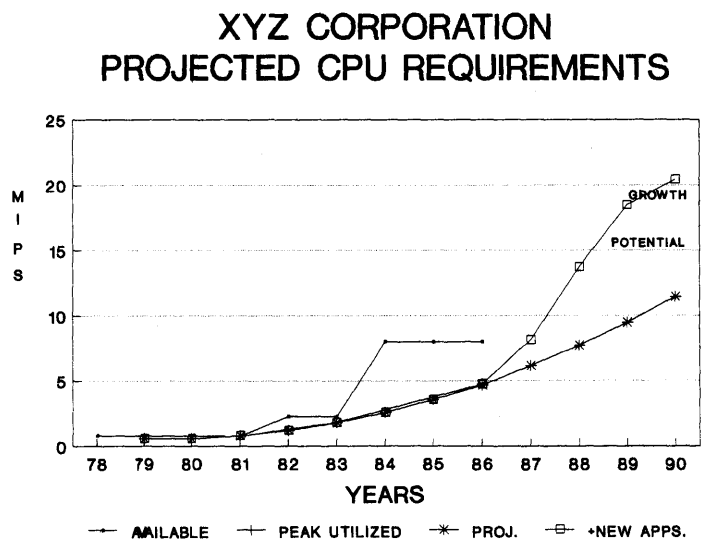


FIGURE-4

3.2 CASE STUDY 2

This corporation has utilized capacity planning techniques for many years; however, no one person was dedicated to the planning and reporting process.

The corporation employs a short-term upgrade policy. This policy was adopted as a method of controlling data processing growth through controlled resource availability.

The technical personnel have been providing management with the monthly CPU utilization report shown in FIGURE-5. Notice that the chart only depicts the CPU growth requirements in terms of the currently installed processor. The

chart does not provide an easily identifiable managerial method of predicting upgrade paths. This deficiency often forced management to pick upgrade paths that have led to less than one-year solutions, thus invalidating their fiscal data processing budget on several occasions.

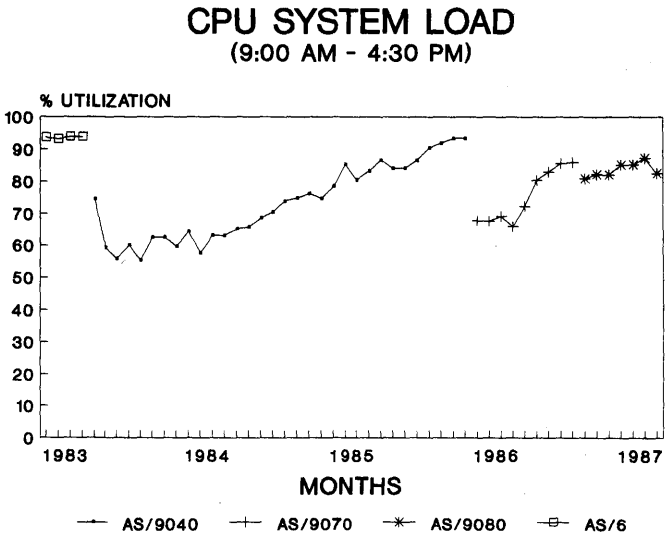


FIGURE-5

After recommendations from the computer vendor and an outside data processing consulting firm, the technical personnel are now producing the monthly CPU utilization report shown in FIGURE-6. Notice how the current chart attempts to normalize the data processing CPU utilization units in a manner that provides the administrative staff with predicted growth information.

By using the chart in FIGURE-6 and a minimal amount of information about the selected computer vendor's hardware, the management team is now better positioned to determine annual CPU growth requirements and processor upgrade paths. This prediction ability gives the management team better control over their CPU budgeting process, thus controlling the requirement of having to "go to the well" multiple times in a fiscal year.

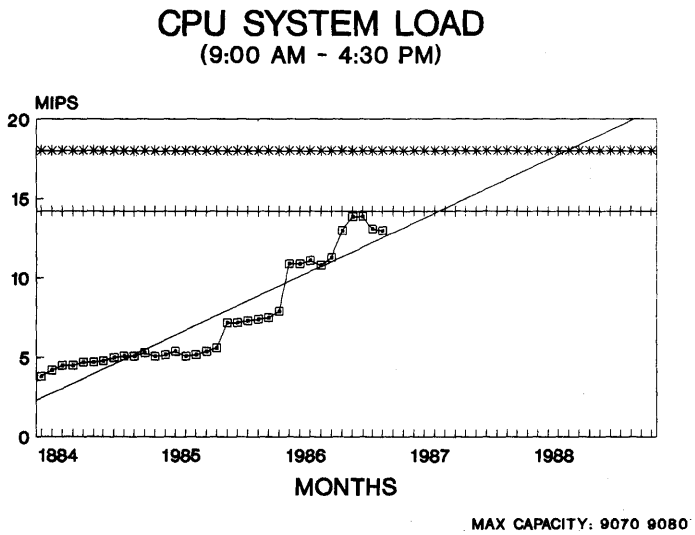


FIGURE-6

3.3 CASE STUDY 3

This corporation had never attempted capacity planning. The philosophy of this corporation was to saturate their processing environment and then purchase a new CPU when the users began to complain loudly. This philosophy was predicated by consistently tight budgets enforced through multiple external managing organizations.

In an attempt to force the managing bodies to realize the true processing needs of this organization, the capacity requirements chart in FIGURE-7 was produced. The basis of this chart was some very sketchy historical data and a series of verbal inputs from the systems users as to their future system requirements and projected CPU resource requirements.

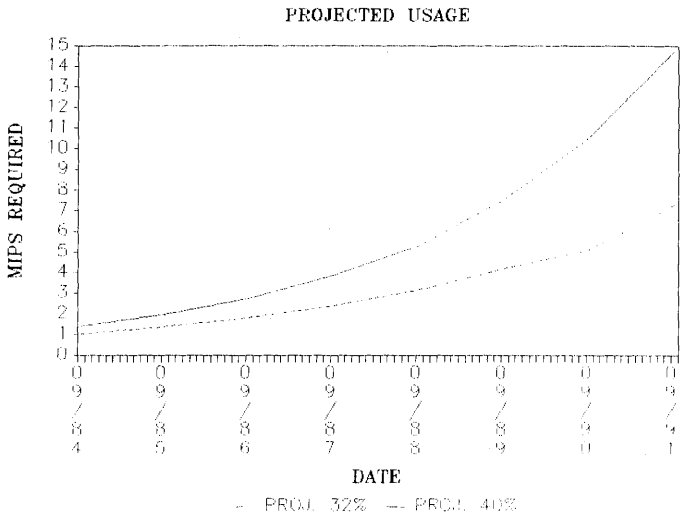


FIGURE-7

The inputs led to two growth scenario patterns that were plotted in FIGURE-7. The data center manager then went to his managing body with this chart and explained the growth requirements. As a result of this action, the data center was given money to proceed with a CPU upgrade, built on the conservative normal growth curve.

The managing body then asked the data center manager to track the current CPU utilization and report back on a regular basis. As a result of this management request, the report in FIGURE-8 was designed. Notice how FIGURE-7 is utilized as a

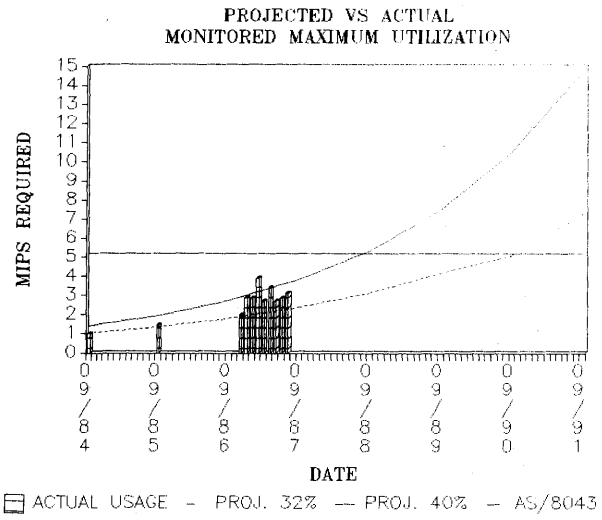


FIGURE-8

base mutually agreed upon capacity projection chart, augmented by an easily identifiable actual peak utilization function.

This tracking process has already assisted the managing body of this data center to understand the peak load CPU processing requirements. In addition to this understanding of processing requirements, the report has instilled a high level of confidence in recommendations made by the Operations Manager.

The ability of FIGURE-8 to track growth will assist the data center management in justifying future CPU acquisitions, thus providing a highly effective, ongoing capacity planning tool.

3.4 CASE STUDY 4

This company was in the process of evaluating upgrades to a two-processor environment, one of which was running a VM and DOS/VSE environment, and the other processor a native DOS/VSE environment. The response time on the VM and DOS/VSE machine had reached extremely poor levels.

A study of this environment was done using the KREUTER technique.[2] Kreuter reports are easily read by both technical and non-technical management after a brief set of ground rules are established.

Some of the ground rules for this study were as follows:

1. All decisions are made on observations rather than theory. The observations were made through a combination of VMAP and DOS/PT data.

2. The following rules of "thumb" were employed for each of the devices measured.

a. CPU capacity limit uniprocessor-90%
b. Channel capacity limit-25% each
c. Control unit DASD capacity limit-25% each
d. Device capacity limit-15% each.

3. All observations were made during peak periods of utilization.

4. All capacity projections were built around the concept of an average peak period derived from at least ten (10) normal processing days of data.

5. Annual growth rate was a number determined by the installation's management.

6. The technique is independent of any operating system environment.

7. The analysis of data looked for items that approached or exceeded the agreed upon rule of thumb numbers printed above. A review of that resource was then made in more detail to determine if there was a bottleneck to the system caused by this resource.

8. The analysis of data was done around an average peak period developed from 10 days of observations.

9. The matrix derived for analysis was cleaned of any anomalies and the results of these ignored in the averaging process.

10. MO used = (growth potential/annual growth) * 12. This figure does not account for compounded annual growth rates and is fairly accurate for the short-term predictions; however, it is the opinion of the author that long-term projections may be suspect.

The CPU and DASD management report for this installation is shown in FIGURE-9.

ON-LINE PEAK PERIOD CAPACITY SUMMARY

RESOURCE	BASE USE	CAPACITY LIMIT	GROWTH POTENTIAL	ANNUAL GROWTH	MONTHS LEFT
CPU					
AS/6620	20%	90%	350%	27%	155 MO
IBM 4341	72%	90%	25%	27%	11 MO
DASD					
CHN AS/6620	22%	100%	355%	26%	163 MO
CHN 1	1%	25%	2400%	26%	1107 MO
CHN 3	2%	25%	1150%	26%	530 MO
CHN 4	8%	25%	213%	26%	98 MO
CHN 5	11%	25%	127%	26%	58 MO
DASD					
CHN IBM 4341	53%	100%	89%	26%	41 MO
CHN 1	13%	25%	92%	26%	42 MO
CHN 3	14%	25%	78%	26%	36 MO
CHN 4	14%	25%	78%	26%	36 MO
CHN 5	12%	25%	108%	26%	49 MO
DASD CTL	75%	100%	33%	36%	15 MO
AS/7350(32)	259%	420%	62%	26%	28 MO

FIGURE-9

The above report led management to conclude that there were enough sources available to run their data center for at least another year. The technical support group was charged with the task of evaluating and tuning the DASD subsystems.

This management directive led the technical personnel to produce the report in Figure 10 showing the activity experienced against the heaviest utilized DASD subsystem.

The report in FIGURE-10 showed extremely high utilization by one control unit and specifically against the VM page and spool volume associated with that control unit.

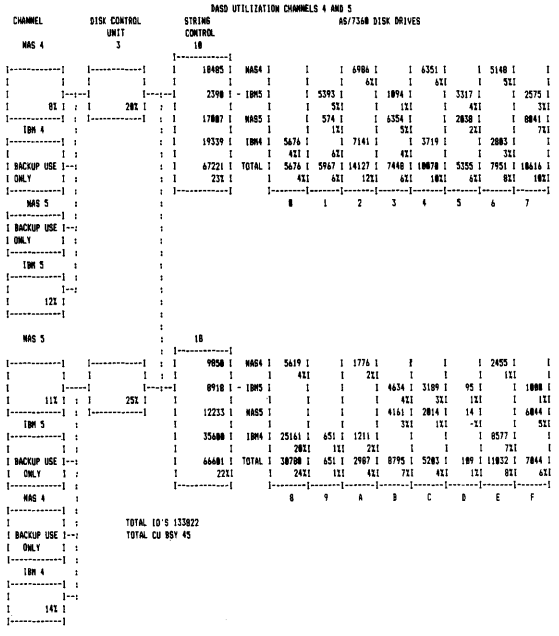


FIGURE-10

The technical personnel then began to look at the VM system's paging functions and determined that a 4-megabyte machine was insufficient for their workload and recommended to management that an additional 4 megabytes of memory were required for the 4341 processor.

The memory was purchased and the system response time improved dramatically. Additionally, the DASD activity against the most active control unit was reduced significantly through the elimination of paging I/O. The account also gained back between 10 and 15% of his 4341 processor cycles. These efforts have allowed the account to postpone a major CPU upgrade for approximately 18 months.

4.0 SUMMARY

Today's performance analysts and capacity planners have an abundance of technical measurement data available. This data is provided by both vendor supplied operating systems and second party monitoring products. The major challenge to the performance analyst or capacity planner is the reduction of this mountain of technically oriented data into simple, clear, management oriented, decision-making documents.

The techniques presented in the preceding case studies are intended to be starting points for the experienced and "green" capacity planner. These starting points are intended to reduce the often long and frustrating reporting cycles experienced when making management recommendations. The key to all of the examples is simplicity. A document that is self-explanatory will direct non-technical management to sound, effective data processing decisions. When this happens, the performance analysts and capacity planners have done their job effectively.

5.0 REFERENCES

1 Gerald F. Lieberman, Quotes For Speakers, (Doubleday & Co. Inc; 1985), p.90 .

2. Robert J. Hardman, "Implementation of a Capacity planning Methodology", CMG TRANSACTIONS WINTER 1986, p.55-60.

COMPUTER ACQUISITION AFTER THE TAX REFORM ACT OF 1986: EXPLORING THE POSSIBILITIES

Sudhir P. Amembal
Shawn D. Halladay
Amembal & Isom, Lease Education & Consulting

Everyone by now is, to some degree, familiar with the major provisions of the Tax Reform Act of 1986. Yet, even as we enter the second year of this legislation, there is still confusion regarding both the application and consequences of certain aspects of the law. Because of this confusion the full impact of the Act is yet to be seen. The leasing of computers, however, will continue in popularity as portions of the new law such as alternative minimum tax and the mid-quarter convention are added to the many other nontax reasons to lease.

The changes in the law, as they relate to computer leasing, can be broken down into four basic areas. The first is the change in tax rates from 46 to 34 percent, effective July 1, 1987. This change alters the aftertax cost of debt for lessees, as well as affecting lease rates. As a result, costs of acquiring computers have gone up. To the extent a lessor's borrowing costs are different from the lessee who is acquiring the computer the lease versus buy decision will be impacted.

The second, and most obvious in its impact, is the repeal of the investment tax credit, effective December 31, 1985. There is no longer that major government incentive to purchase and own new computer equipment. Potential acquirers of computer equipment will look at leasing more closely, now that the major impetus for buying the equipment has disappeared. From a quantitative perspective the cost of buying in lease buy decisions will have gone up substantially, buy leasing costs will also increase. Another result of the repeal of ITC will be a surge in the used equipment market. Without the government subsidizing purchases of new equipment used computer leasing becomes a more viable alternative.

The Alternative Cost Recovery System for tax depreciation has been modified, effective December 31, 1986. The modifications include changes in class lives, rates and methods of calculating the deduction. There have been shifts in depreciation lives for certain types of equipment, but none which affect computer leasing. This is not to say that there are not any depreciation issues affecting the acquisition of computer equipment. The depreciable life of a computer is still classified as five years but, due to a change in computing the yearly depreciation to 200% declining balance, it must now be depreciated over six years instead of five. Interestingly enough, although the period was lengthened, the actual depreciation was accelerated somewhat. Under the old law, 37% of the computer equipment could be written off during the first two years but, under new law, 52% can be written off. Again this type of change affects the buy side of lease versus buy, and also the lease side as lessors adjust their rates downward to reflect these changes.

A new twist in the law called the mid-quarter convention certainly has an impact on computer acquisition. Under this convention, if a computer user acquires more than 40% of its equipment (this includes equipment of any type acquired by the corporation) in the last quarter it will have to use a different depreciation method. This method, not only becomes a bookkeeping nightmare, but also lowers the value of depreciation in that year, is a definite tax disincentive.

This mid-quarter convention, however, only applies to equipment which is purchased, not leased. It is plain to see, then, that leasing provides a mechanism through which the lowering of depreciation value can be avoided. By properly matching their own and lessor year ends so as to avoid those lessors who themselves are in the mid-quarter penalty, a lessee can obtain the most favorable rates. This is another factor to be considered in the lease versus buy decision and while, perhaps, not causing everyone to lease, it will certainly cause acquirers of equipment to again consider leasing more favorably.

All aspects of the new tax law impact the lease versus buy decision, but none is as difficult to comprehend and apply as the alternative minimum tax ("AMT"). The concept of the AMT is that all corporations will pay at least some minimum level of tax. While simple in theory, this provision represents one of the most complex and pervasive changes in the law. The tax itself is based upon regular taxable income plus certain preference items. This total is multiplied by the AMT rate of 20% to derive the tax (preference items are certain items used to reduce taxes, the nature of which are often times thought of as "tax sheltering" devices). Many corporations may not owe regular taxes because of their deductions but, after adding back their preference items to regular taxable income, they most likely will owe an alternative minimum tax. Because the greater of the AMT or the regular tax must be paid it is plain to see that a corporation would want to avoid being an AMT taxpayer.

The two preferences of major concern to acquirers of computer equipment are the preference on accelerated depreciation and the pretax book income preference. These preferences may be explained as follows:

1. Depreciation Preference. Any excess of the regular accelerated depreciation (200% declining balance) over 150% declining balance must be added back to regular taxable income as a preference. This reduces the benefits of depreciation to those users who purchase computer equipment.

2. Pretax Book Preference. One half of the excess of a corporations's pretax book income over its AMT income must also be added back to regular taxable income as a preference. This preference affects both purchasers and lessees albeit differently for both.

Lease versus buy under AMT becomes incredibly complex as now the potential lessee must know exactly what its AMT position is, and then integrate it into the analysis. Many acquirers, however, will find it advantageous to lease because the leasing of equipment does not create the depreciation preference. Leasing, in effect, become a means to avoid AMT or to bring a lessee out of an AMT position more quickly. In this respect leasing can be expected to grow as lessees take advantage of this tax planning alternative.

Corollary to the aforementioned advantage, lessees will also be able to shop around for better rates because of the effect of AMT on lessor pricing. Since AMT lessors pay more taxes, their rates should theoretically be higher. It would behoove lessees to seek out non-AMT lessors through appropriate rate comparisions.

All things considered, the Tax Reform Act of 1986 has created opportunities for computer equipment acquisition through leasing. The opportunities take the form of direct lessee advantages, such as avoiding the AMT through leasing, and indirect advantages, such as the impact of the AMT on lessor pricing. Those acquiring computer equipment should carefully study the ramifications of the new tax act and then take advantage of the opportunities which it offers.

INCORPORATING EXPERT SYSTEMS TECHNOLOGY INTO SOFTWARE PERFORMANCE ENGINEERING

Major Doyle J. Weishar
U.S. Army Information Systems Engineering Command

ABSTRACT

Software performance engineering is the process of considering performance goals when software is initially designed in order to achieve maximum effectiveness at minimum cost. Software performance engineering is traditionally conducted by assigning a performance analyst to conduct a study of a designated system under development. In the United States Army Information Systems Engineering Command (USAISEC), we are embarking upon a new approach: we are striving to place the tools and techniques into the hands of the applications software developers. In this paper we describe our efforts to develop the intelligent tools needed to transfer the domain expertise of a performance analyst to a non-performance oriented person.

1. INTRODUCTION

In the late 1970s researchers in the area of artificial intelligence began to realize that the problem solving power of a program comes from the knowledge it possesses, not just from the formalisms and inference schemes it employs. This realization led to development of special-purpose computer programs that came to be called expert systems [1]. Since that time expert systems research has received much notoriety. Expert systems technology has been used to solve problems in many disciplines. The most notable expert systems applications have been in the fields of chemistry, geology, medicine, and computer systems. This paper describes the entrance of expert systems technology into yet another field: software performance engineering. We first offer the reader a short description of software performance engineering. From there we describe how software performance engineering is conducted within the United States Army Information Systems Engineering Command (USAISEC) and then what motivated us to venture into the expert systems area. Finally, we describe the new tools that we are developing in order to foster software performance engineering within USAISEC.

2. SOFTWARE PERFORMANCE ENGINEERING

Software performance engineering is the process of considering performance goals when software is initially designed in order to achieve maximum effectiveness at minimum cost. Implementation of software performance engineering begins early in the development of a system and continues throughout the system life cycle. During the requirements definition and preliminary design stages, the performance analyst checks the feasibility and desirability of the functional architecture. In the later stages of the software life cycle, software performance engineering concepts are used to study algorithms, monitor progress toward meeting performance goals, and assess the impact of design changes [2].

Software performance engineering offers many benefits for system development. Among the most notable benefits include: the capability for early evaluation of design alternatives; the ability to isolate potential system bottlenecks and assess system performance; it results in increased productivity by making better use of the resources that would have been required to fix problems later; and finally, software performance engineering improves software quality since performance is an important aspect of software quality that is often overlooked [2]. An in depth discussion of software performance engineering is beyond the scope of this paper. Significant references in this field are [2], [3], and [4].

3. SOFTWARE PERFORMANCE ENGINEERING IN USAISEC

The charter for the software performance engineering program lies within USAISEC in the Systems Performance Division of its Quality Assurance Directorate (QAD). QAD has traditionally conducted software performance engineering by assigning a performance analyst to conduct a study of a designated system under development. In the conduct of his (her) study, the QAD analyst will collect pertinent data, build a model using those data, and apply the principles of software performance engineering in the analysis of the modeled system. To date, QAD has analyzed a number of systems. These include financial, logistical and personnel systems. In USAISEC there are a great many systems in development -- many more than the three to four performance analysts assigned to the Command can effectively analyze. When we in QAD began looking at this problem, the obvious solution was to hire more analysts. However, this was not an option. Therefore, we were required to look for new ways in order to foster an active software performance engineering program. One of the actions that we took was to publish the Command's Software Performance Engineering Handbook [5]. The handbook identifies tools and techniques for implementing software performance engineering. It is oriented towards systems developers (rather than performance analysts). Despite this, our feedback has been that the developers tend to believe that these concepts and methodologies are difficult to learn, are of questionable value, and take them away from their primary function which is to develop systems. To overcome this recalcitrance, we are incorporating expert systems technology into our program. Our intent is to offer software developers easy to use tools that will ingest the performance analyst's domain expertise into the development process in a painless manner. In the remainder of this paper we describe our efforts to develop these tools.

4. BLDR: AN AUTOMATED CRYSTAL MODEL GENERATOR

Modeling is the cornerstone of a software performance engineering analysis. Modeling ranges from pen and pencil exercises to very sophisticated simulations. With the possible exception of the pen and pencil exercises, modeling of a computer system is a very complicated task. Our performance analysts are capable of constructing the entire range of models, but most often construct analytic models. While analytic models are not as precise as simulations, they have been shown to be quite effective for modeling most computer system applications [6]. We use a tool developed by BGS Systems to construct our analytical models. This tool is called CRYSTAL. CRYSTAL has been developed specifically for modeling new software applications. This tool eliminates much of the tedium involved with analytic model construction by eliminating the need to code the underlying mathematical equations necessary to solve the queueing model. Despite this, there is a

reasonable amount of performance analyst knowledge required to use the tool effectively. As stated earlier, we did not want to require systems analysts to spend a lot of time learning the performance trade. But since modeling is so important, we had to think of a way to easily place the powerful modeling tool in their hands. We will do this with a tool called BLDR.

BLDR is essentially a preprocessor to the actual CRYSTAL analytical modeling routines. BLDR takes screen input from a user and constructs a model that can be read into CRYSTAL. The user is not required to have any performance background, only an idea of the hardware and software environment in which the application is to be run, and an idea of how the proposed application works. The following is a subset of a typical BLDR/user dialogue:

How many disk drives will you be using? 8
What type? STC 8380
List the names of the logical files and the logical record lengths you will be using? Finance_Master_File, 80, SSN_File,128

Once the user has completed his (her) description of the new application and its environment, BLDR automatically constructs the CRYSTAL model. We have tailored BLDR to suit our particular environment. Therefore, many system parameters do not need to be input by the user. We are trying not to make this a cumbersome tool to use. We are implementing BLDR in PASCAL/VS. It is approximately 50% complete. In its current version it will construct models of simple applications. We hope to have it building detailed models soon.

5. EXPERT SOFTWARE PERFORMANCE ENGINEERING

When BLDR is complete it will be an effective and easy to use tool. However, BLDR alone does not satisfy our goal -- a user still has to know performance to use the CRYSTAL tool. Therefore, we are building two tools that are designed to place software performance engineering expertise into the hands of our software developers. Our tools are called QAXPRT and DBXPRT. QAXPRT is a mainframe tool used in consonance with BLDR to assess overall system performance. DBXPRT is a personal computer based tool that is used to assist in database design. The next two sections describe these tools and highlight our progress in their development.

6. QAXPRT

QAXPRT is a rule-based expert system being implemented in PASCAL/VS. It consists of a collection of domain knowledge, called the knowledge base, and general problem-solving knowledge, called the inference engine. The knowledge base contains facts (data) and rules [1]. The rules express a conditional, with an antecedent and a consequent component. The facts express assertions about properties, relations, and propositions [7]. The rules and facts in our knowledge base have been formed using our collective expertise and guidelines in the performance literature. The rules use those facts as the basis for decision making. We use IF THEN constructs to represent our rules. An example of typical facts and rules in QAXPRT is as follows:

FACTS: User requires response time <= 3 secs.
Workload X is in application of user.

RULES: IF workload X response time > 3 secs
and > user_defined_limit
THEN warn user and look for cause

When the current problem situation satisfies or matches the IF part of a rule, the action by the THEN part of the rule is performed. In our example, if workload X's response time is greater than the user's acceptable limitation, then the system will notify the user and try to find a cause.

The second component of our expert system is the inference engine. The inference engine works by the matching of rule IF portions to the facts. We use a method of backward chaining by which inferences are formed from successive execution of rules.

QAXPRT assesses the overall performance of the application modeled by BLDR. After the BLDR model is evaluated by CRYSTAL, QAXPRT scans the user.CRYSTAL.LOG file and highlights possible performance problems. Recalling our earlier example will illustrate how QAXPRT works. In that example, QAXPRT notices that response times are above the user defined acceptable level. After notifying the user, QAXPRT begins looking for a cause. It is likely that QAXPRT may discover that certain disk devices are overly utilized. QAXPRT would infer this by matching the following fact and rule.

FACT: A disk is overly utilized if its utilization > 40%

RULE: IF disk X U > 40%
THEN redistribute files

Knowing that a disk is overly utilized, QAXPRT would begin to search for underutilized disks. Upon finding a candidate (disk Y), it will recommend to the user that one or more files residing on disk X be shifted to disk Y. The user then takes this recommendation and inputs the change through BLDR to CRYSTAL for reevaluation of the model.

As mentioned earlier, QAXPRT is being implemented in PASCAL/VS. Our knowledge base is over 50% complete, but our inference engine is still in the design phase.

7. DBXPRT

DBXPRT is a rule-based assistant for database design. It is being implemented in PROLOG on a personal computer. The conceptual framework of DBXPRT is the same as QAXPRT (i.e., it consists of a knowledge base and an inference engine). DBXPRT differs, however, in its intended use. Instead of assessing performance for an entirely new application, DBXPRT is being developed to be able to predict the change in loading that would result from adding a proposed transaction to an existing application. Specifically, DBXPRT concentrates on diagnosing potential performance problems associated with queries to large databases.

We have developed a prototype system that is able to predict response times associated with various types of queries (keyed, non-keyed, sorted, etc.) to different types of databases (VSAM, ISAM, sequential, DBMS, etc.). While there is much to be completed, we can see the light at the end of the tunnel with DBXPRT. Despite being only a simple prototype system, it has already demonstrated the ability to win arguments with "experts" by correctly predicting results that at first sight appeared to be counterintuitive.

8. CONCLUSIONS

In this paper we have shown how expert systems technology can be incorporated into the field of software performance engineering. We have shown that an automated model builder can be used in conjunction with a commercial modeling package and an expert analyzer to analyze the design of software, and assess the performance of that software. While there is much work to be completed, we feel that definite progress is being made. Ultimately, we will achieve our goal of providing easy to use intelligent performance engineering tools to our software developers.

REFERENCES

[1] Waterman, D. A., A Guide to Expert Systems, Addison Wesley, 1986.

[2] Berry, M. E., "The Best of Both Worlds: An Integrated Approach to Capacity Planning and Software Performance Engineering," Proceedings, CMG XV, San Francisco, 1984.

[3] Smith, C. U., "The Prediction and Evaluation of the Performance of Software from Extended Design Specifications," Ph.D. Dissertation, University of Texas at Austin, 1980.

[4] Smith, C. U., "Increasing Information Systems Productivity by Software Performance Engineering," Proceedings, CMG XII, New Orleans, 1981.

[5] U.S. Army Computer Systems Command, Quality Assurance Directorate, Software Performance Engineering Handbook, Fort Belvoir, VA, 1984.

[6] Morris, M. F. and Roth, P. F., Computer Performance Evaluation: Tools and Techniques for Effective Analysis, Von Nostrand Reinhold Co., New York, 1981.

[7] Rich, E., Artificial Intelligence, McGraw Hill, New York, 1983.

THE GOLDILOCKS SYNDROME

Mike Berry
Amdahl Corporation

ABSTRACT

For some time now I have admired the medical profession's use of the word syndrome. Once they have attached some named syndrome to a set of symptoms and manifestations, they create an incredible shorthand way of communicating a related set of issues and problems, and where one exists, identify a solution. Now, the simplest of phrases, e.g., Downs Syndrome, Tourette Syndrome, conveys a book's worth of knowledge to the listener, and assuming the listener is a similar professional, the partnership can expend all their energies on the solution. What, I wondered, could we do in Data Processing (DP), a scenario that abounds in issues and problems, and always a plethora of solutions.

Another "coding" scheme that has long since caught my attention is Richard Nolan's Stages of Growth theory. As well as being convinced by the content of Nolan's theory, I am very impressed with the "lifetime" nature of this theory, i.e. Nolan started with Stages I to IV, then told us about Stages V to VIII, and then comes IX to XII, and so on.

So, I thought, if I could invent DP "syndromes", a) I can bring the power of the "shorthand" system to our dialogues, b) I can start a series that will see me through to retirement, and c) while I cannot put "Berry" to all my syndromes (as the medical profession does), I can become famous just like Nolan. The fact that I am here today tells you all you need to know about how famous I am--most of you never heard of me. However, the syndrome thing is working--elsewhere I have described the following syndromes.

- the Shredded Wheat Box syndrome
- the Condensed Milk syndrome
- the Infinite Capacity syndrome
- the Four-Eyes syndrome

AND NOW -- THE GOLDILOCKS SYNDROME

This is a lighthearted, but deadly serious examination of how Data Processing (DP) can change the way technology is promoted, quantified and valued. I want to focus particularity on the way capacity is determined, dispensed and valued. In doing that I seem to cast some doubt on practices sometimes seen. The extremes to which I point are deliberately overstated. I want you to leave here today with the syndrome firmly fixed, then from hereon we can use the power of that "shorthand" to quickly identify the problem and get to the solution. So, if you are in capacity planning and capacity management, go with the extremes. I've been in capacity management too, and I am here as a friend, and here to help!.

The core of this presentation is to do with how we have traditionally brought technology (i.e. capacity) to the client, why we now need to change the way we do that and how to change it. BUT FIRST.....

WHAT DOES THAT HAVE TO DO WITH GOLDILOCKS???

I am going to develop an analogy between the way capacity planning is sometimes done (NOT OF COURSE, BY ANY OF THE ORGANIZATIONS REPRESENTED IN THIS ROOM), and the "Goldilocks and the Three Bears" story. There are no prizes for guessing that Goldilocks is the client, the Bears are the Capacity Planners and the Bear's house is the Data Center. Incidentally note if you would, that I use the word "client", not "user". As far as I know, DP and the drug-pushing business are the only ones that call their customers "Users." DP stands for "drug pushing"??? Let's start calling them "Clients".

My first question is "Were the Bears too possessive?" Why go out for a walk leaving the house open if you don't want anyone to wander in perhaps to use the facilities. By the way, from now on, since I gave you the capacity planners, data centers, clients analogy, any pictures that appear in your head are yours, not mine!

Do we get too possessive about the capacity we plan? Do we almost become irritated when someone uses it? Even if we do not become irritated, do we watch them as they use it? Do we have some notion that there are "good" ways to use capacity and that clients do not know how: Do we WATCH them.

My next question is, "What example were the managers (Mama and Papa Bear) setting for Baby Bear? Did he/she behave that way from their example? Could the Bears have achieved much more out of the situation? Any lessons for DP here? Remember, any pictures running through your heads, any "Bears" and "Goldilocks" you are identifying are your responsibility.

ARE WE TOO POSSESSIVE IN DP?

DO WE PUSH TECHNOLOGY FOR TECHNOLOGY'S SAKE?

DO WE SEE CAPACITY AS OUR SANDBOX?

IS THE CAPACITY "OURS" OR "THEIRS"?

HOW DID ALL THIS COME ABOUT?

What has been our traditional approach to what technology (capacity) we do need, who can best select it, who needs to watch how it is used? Traditionally we have assumed that only we in DP can select technology (capacity). Only we know how to use it well. We need to watch the client's use of it. Clients do not know enough to be "accountable".

THE RESULT OF THIS STRATEGY?

TO OUR CLIENTS--WE LOOK LIKE POLICEMEN!!

Most people would see being a policemen as "not a nice job". The capacity planners in the audience--did you know that was what you were signing up for? The capacity managers in the audience--did you tell the new recruits what the job was really about?

DO YOUR CLIENTS SEE YOU AS "GUARDING" THAT CAPACITY, DISPENSING "APPROVED USAGE", "ALL VIOLATORS WILL BE PROSECUTED?"

Let's spend a few minutes looking at the kind of words we use when we set annual objectives for our capacity planners and resource managers. (You do set objectives, don't you?) "CONTROL-CONSTRAIN-RECLAIM-MONITOR-IDENTIFY HOGS, etc." i.e. "POLICEMEN" like words.

If you do nothing else when you get back to the workplace--look at the words that are used in the annual objectives of people in Revenue Center or Profit Center occupations. I guarantee you will see words like "PROMISE-GROW-INCREASE-SERVICE" -- all proactive "do-more-of-it" words.

IN OTHER WORDS -- SELL!

Let me come back to the concept of selling. First let's look at another traditional approach we take in DP when delivering (explaining) technology (capacity). Don't we all too often, buzzword/acronym our client to death? Don't you sometimes get the feeling (like a young bride might get on her honeymoon) -- "never mind the technicalities, just let me enjoy it"?

Let's return to this notion that we can "sell" our capacity rather than "police" it. The selling process implies "seller", "buyer", "price" and "value". How can we best put a value on capacity?

Let's ask a few questions.

- "Who knows most about,
  - Technology
  - The Business
  - Organizational Values
  - Corporate Budget
  - LRP?

THE CLIENT EVERY TIME -- Except technology. (We hope we know technology or we are in trouble.) But the client certainly knows the rest. Does this give us a lead to how we might manage this "supplier/customer" relationship? Isn't what we are looking at the classical supply/demand situation?

Simple supply and demand makes sure that the price fits the demand, the price stays right, whatever right means, and also has a fundamental concept. And, that is--that while we might start off thinking price equates to cost, it does not have to.

YES FOLKS, WE ARE TALKING ABOUT BILLING FOR SERVICES!!!

WHY? o being a policeman means being a worrier.

- Now what's going on?
- Where shall I look for rogues and vagabonds next?
- What on earth are they up to now? Etc.
- Guess What! We in DP do not know all we need to know to worry at our best.
- But, we insist on doing it. We insist on being the organization's "conscience".
  - Are we/they doing it right?

THE CLIENT DOES!

PLACE THE ACCOUNTABILITY WHERE THE VALUE CAN BEST BE DETERMINED. NOW DP IS SEEN AS A SUPPLIER OF VALUED SERVICES -- NOT AS A NECESSARY EVIL.

NOW, BACK TO THOSE BEARS.....

HOW COULD WE HAVE HELPED THEM?

IT'S CLEAR, ISN'T IT--THEY SHOULD HAVE OPENED A HOTEL!!!

So, DP isn't that simple, but we can start encouraging our capacity planners/resource managers to look outwards/not inwards. Worry less about what's going on in the system, more about what's going on in the business. Provide a mechanism for valuing the services. BILLING.

We can bill for services, but:

- Keep It Simple
- Keep It Consistent
- Encourage "Good"
- Discourage "Bad"
- Probably Just CPU/Storage/Printing/ Terminals

Encourage Capacity Planners/Resource Managers to get out into the business community--NOT JUST AT BUDGET TIME.

Change the tone of Objectives/Measurements.

Make VALUE the watchword. And, above all.....

SELL - SELL - SELL !

System Performance Management and Capacity Planning

Dr. Bernard Domanski ***
Bell Communications Research, Inc.
Piscataway, New Jersey 08854

John P. Pilch

Bell Communications Research, Inc.
Piscataway, New Jersey 08854

Abstract

This tutorial is designed to introduce the manager/analyst to a variety of techniques and methodologies for capacity planning and performance evaluation. In particular, the system life cycle model, benchmarking, modeling techniques, workload characterization, and performance management will be discussed. While no new results are presented here, this tutorial will provide an introduction for the novice and an overview for the manager.

The presentation will be divided into three sections. They are:

1. A General Systems approach to Performance Management and Capacity Planning.
2. Workload Characterization and Prediction Techniques.
3. Simple Performance Calculation.

This paper is intended to expose some of the many facets of Computer Performance Evaluation (CPE), and augments the technical content of the presented sessions.

1. Introduction

Large computer systems incorporate many features such as multiprogramming, database management, virtual memory and complex operating systems. These features make performance management and capacity planning of these systems particularly difficult. The purpose of this paper is to outline those aspects of capacity planning and performance management that are the most critical in viewing a complex computer system. Topics to be covered include the system life cycle model, benchmarking, modeling techniques such as operational analysis and queuing theory, simulation, workload characterization and performance management. While none of the topics will be covered in great detail, enough information and references will be provided so that the reader may peruse topics of interest on his/her own.

2. The System Life Cycle Model

A systems' life cycle can be represented in many ways. Figure 2.1 is one model of the systems life cycle.

THE SYSTEM LIFE CYCLE MODEL

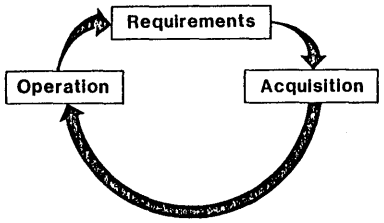


Figure 2.1

Expanding on this model, we note that during the requirements phase we need to understand the role (objectives) of the system, the load the system must support and the service the system is required to provide. During this phase, one must define more clearly the functionality of the desired system.

Once the requirements are identified, we begin to enter a phase of acquiring the needed resources to support the system requirements. Here we usually must compare among a set of alternatives over the entire expected life of the

*** This author is currently on sabbatical leave from the Computer Science department of the City University of New York at the College of Staten Island. Portions of this work were conceived and performed at Bell Communications Research, Inc., Piscataway, New Jersey 08854.

system. That is, as the system matures, the load that it supports will normally increase and the resources required to support this increased load should be identified during the acquisition phase. Thus, a plan should exist to support the system requirements over its expected life cycle. In essence there are two "types" of comparisons going on here:

- For each alternative "vendor" or set of resources, one must compare alternative changes in the set of resources (more memory, disks, etc.) over the life of the required system, thus producing a plan.
- One must also compare alternative "vendor" plans.

After we have compared and selected (obtained) the needed resources, we must operate the system including all the resources acquired to support the system. In essence, we must manage the system so that it meets its required goals in a timely fashion given the load. We, therefore, must manage the set of resources to ensure that the resources provide the required service given the load. This is sometimes referred to as Performance Management.

Figure 2.2 is an expanded view of the System Life Cycle.

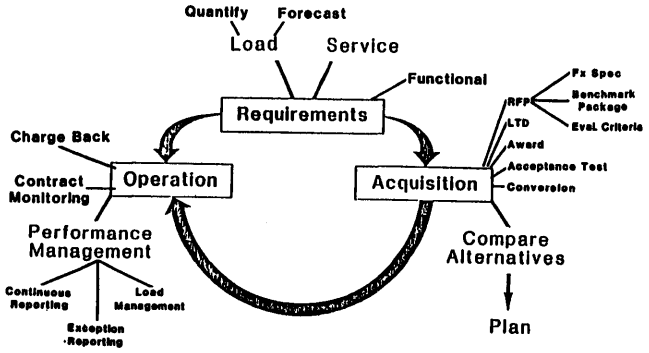
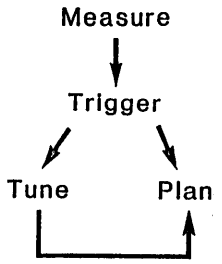


Figure 2.2

Performance Management then consists of measuring the system's service during its operating phase and comparing the observed service measurements against the planned or required service. When the observed differs enough from the planned, some action is triggered. Normally one tends to tune the set of resources in some way first to improve the overall system service. If tuning was successful, one tends to update the initial plan and continue. If tuning was not successful or possible, one tends to enter into a major planning activity usually to acquire additional resources to support the system.

Figure 2.3 represents the Performance Management process.



Performance Management

Figure 2.3

Now during the tuning process one tends to have several alternative optimization approaches suggested by the measurement data. At least conceptually one must compare these choices and select an alternative to implement. Also, during the triggered plan or re-plan process, one tends to have several alternative implementation plans from which to choose. Again, one must compare from among a set of alternatives. To get a larger perspective on the system life cycle, figure 2.4 pulls the major components together. Appendix A contains several diagrams that are referred to in the presentation.

THE SYSTEM LIFE CYCLE MODEL

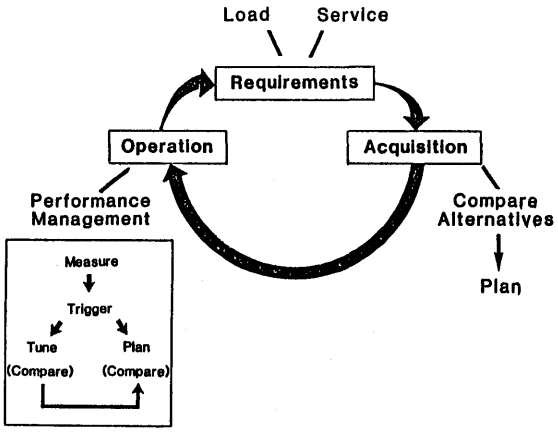


Figure 2.4

3. Benchmarking

A Benchmark is defined by Webster as a point of reference for measurement. The act of obtaining the point or point(s) of reference is defined as benchmarking.

An important aspect of benchmarking is a clear experimental design. One needs to have identified the questions being addressed, the definition of the system boundaries of interest, a representation of the load for which the system is to respond, and a means by which the load can be presented to the system under test and service data collected.

If, for example, we were comparing two processors to answer the question "which processor does the floating point operations more effectively?", then the system boundaries could be just the processor (possibly processor and memory) and the load needed to represent the floating point operations. Let us assume the load consisted of a set of floating point operations, then this set of operations would be presented to each of the processors and one would measure how long it took the set to complete. (See figure 3.1.) The operations might be presented to the system as machine instructions.

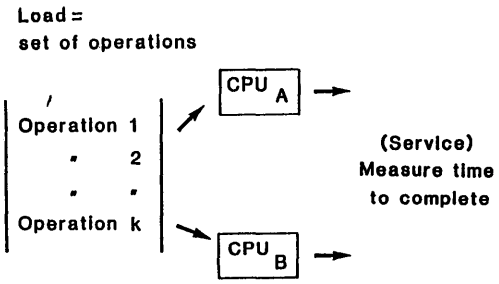


Figure 3.1

If the questions were: which system does the matrix inversion the best? Then the system boundaries change and the representation of the load changes. For example, if the matrix inversion involves using a FORTRAN program to do the matrix inversion, then, we are interested in how efficient the FORTRAN compilers are on the alternative system(s). One would tend to take the original FORTRAN program and have each system generate the compiled object code. Then one would simulate each system to measure the service by timing how long it took for each system's object code to run. The differences in the observed measured data (across system) might be because the hardware or the efficiency of the FORTRAN compiler(s) to generate good object code. As long as the original question was to determine which system did the best FORTRAN-based matrix inversion, the measured data would present a reasonable estimate.

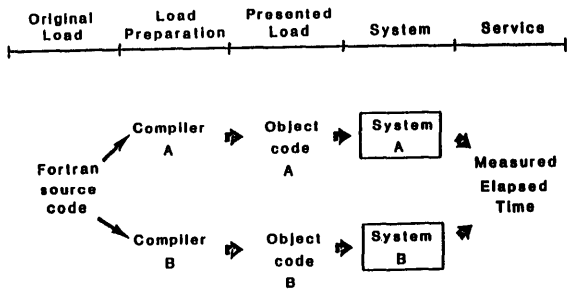


Figure 3.2

As we better understand the work that the final system is to do, we will be more able to represent the load and the means of loading and measuring the system(s). Let us assume that we have the representative workload and we are interested in determining the configuration(s) for a timesharing system. The question becomes: What system configuration will support the load (provide acceptable service) over the system's life?

This leads us to generate a set of subquestions that come from hypotheses about the system being considered. That is, knowledge about the computer system allows us to think of alternative configurations.

Question: What initial system configuration will support the load?

Subquestions:

Initial configuration

- memory needs?
- processing needs?
- storage needs?
- communication needs?
- software needs?

From these subquestions, a set of alternative initial configurations tend to be generated. It is these alternative configurations that we must obtain load-service relationships to compare these alternatives. It is a result of comparing these alternatives that we really develop the computer capacity plan.

Let us assume that several alternative configurations are proposed for evaluation. Figure 3.3 represents key alternatives that must be understood to select the initial configuration.

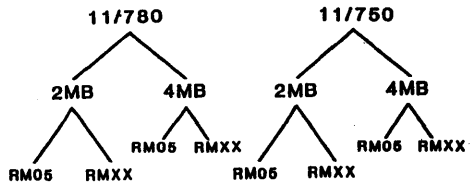


Figure 3.3

That is, to select the initial hardware configuration, we ideally need to compare eight load-service relationships (curves) as partially shown in Figure 3.4. To create the load-service relationships we must collect response time values for selected terminal levels and hardware configurations. From Figure 3.5 we see that for a single representative program that 32 "tests" might be performed to collect the desired data to make the comparison.

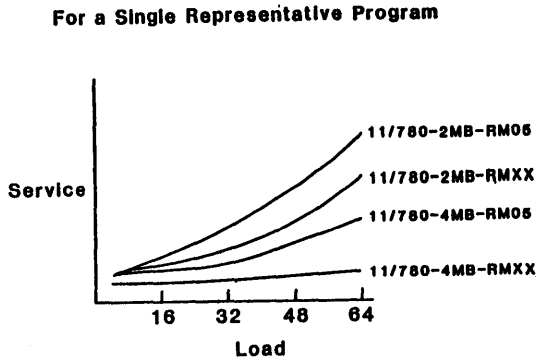


Figure 3.4

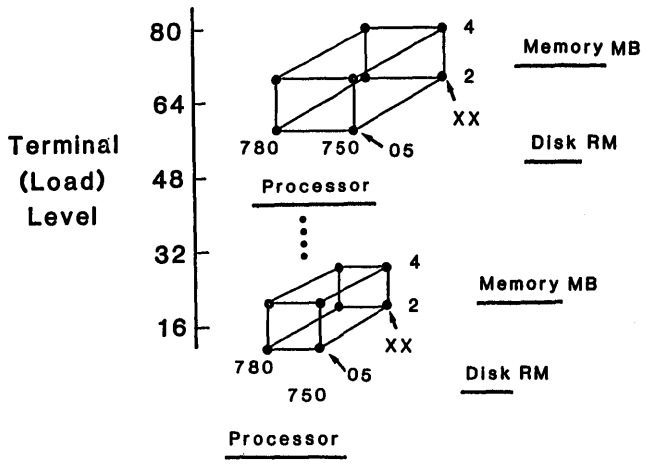


Figure 3.5

If we decide to collect this data from a live system, we might need about 32 hours of test time. We would also need a way to present the load to the system configurations under test and measure the time it took to process the load. This is usually done with an RTE (Remote Terminal Emulator).

Remote Terminal Emulation is an approach to the testing and evaluation of computer systems in which a driver is implemented external to and independent of the system being tested. The driver, called the Remote Terminal Emulator (RTE), applies a specified workload to the system such that it appears to the System Under Test (SUT) that it is connected to live terminals. The objective of remote terminal emulation is to drive a SUT in a way that is controllable, repeatable, economical and realistic. Figure 3.6 is a schematic of the flow of information in a remote terminal emulation experiment.

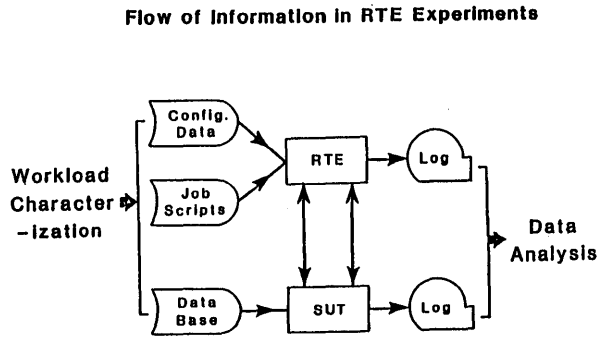


Figure 3.6

In general the workload or representative program(s), etc., must be represented in some way to the RTE so that the system under test (SUT) believes it is the real world presenting the load. This tends to be done by creating SCRIPTS that contain message text and control information. Figure 3.7 is one example of what would be included in a SCRIPT.

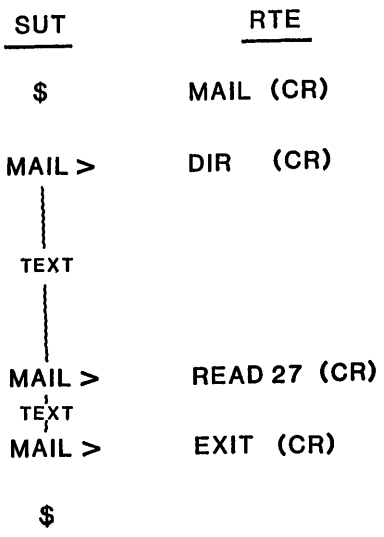


Figure 3.7

In general the SCRIPT has the information about what needs to be sent to the system under test and what it should expect. All the interactions with the SUT are time-stamped and put on a log file for later reduction and analyses. If Figure 3.7 represented a MAIL user and we ran an experiment where there were 16 such users, our RTE log file might look like Figure 3.8 if it were sorted by terminal number and time of day.

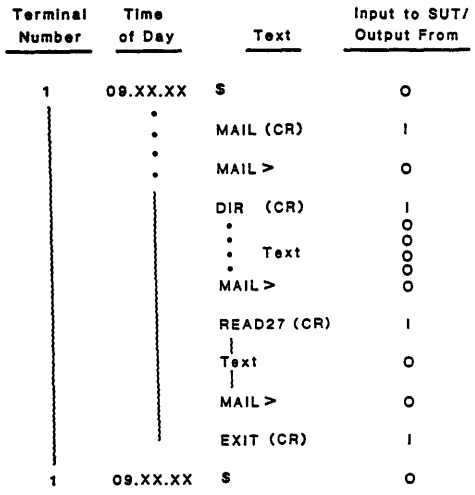


Figure 3.8

Much work has been done using RTE's and the reader is referred to [Domanski, B., "Load Driving A System," Computer Performance, Vol. 3, No. 4, Dec. 1982].

Another approach to obtaining the load service relationship is via modeling instead of benchmarking the actual system. The next section will present an overview of Modeling.

4. Modeling

A model of a system is a representation of the system that consists of a certain amount of information about the *organization* of the system, and is built to to study it. Since different kinds of questions can be asked about a system, several different types of models can be constructed. They all represent the same system, but look at the system for different purposes or they contain varying amounts of detail.

A model can be viewed as a system (here *system* is not an abbreviation of a computer system). Various models correspond to partitioning the system into different components, representing the interactions between components and its operating environment. These conceptual models play a fundamental role in evaluating computer system performance. They consist of all the information the evaluator has about the behavior and structure of the actual system. The deeper the knowledge of a system, the better the conceptual model and the easier and more successful the study is likely to be.

Characterizing the behavior of a system is often done using the concepts of state transitions.

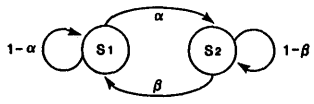


Figure 4.1

The state of the system at time *t* is defined as the values of the system parameters of interest at time t. Any variation in these values is viewed as a transition to another state, i.e., a *state transition.* A state transition model associates *probabilities* with each state transition.

A *simulation* model reproduces the behavior of the actual system over time. Its behavior in time, under stimuli that represents the actual systems environment, is observed, and performance data is collected. In this way, the simulation model is treated just like the actual system. Thus, designing experiments for simulation models is the same as designing experiments for the actual system.

An *analytic* model consists of a set of mathematical equations that are solved by following the evolution of their solutions over time. Computer systems are seen as discrete systems; that is, they evolve by discontinuous state transitions called *events.* These models include equations expressing the logical conditions under which these events occur.

Queuing Concepts and Operational Analysis

Queueing models are popular analytic models. As an example to illustrate the concepts underlying queuing models, consider a barber shop.

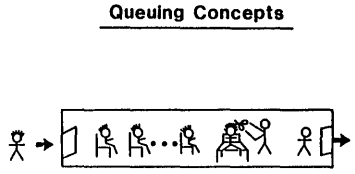


Figure 4.2

Here customers enter the barber shop, and wait until the barber is available to cut their hair. It takes some amount of time to cut their hair, after which they leave the barber shop. This is known as a single server queuing system, and is formally denoted by the following diagrams.

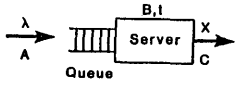


Figure 4.3

Problems analyzing these systems are all derived by making assumptions, which are testable, about the queuing model. This simple approach is called *operational analysis,* developed by Buzen & Denning (The Operational Analysis of Queuing Network Models, Computing surveys, Vol, 10, No. 3, September 1978). Appendix B contains definitions and basic formulas used in Operational Analysis.

In both the benchmarking and modeling approaches to obtaining load-service relationship there is a need to characterize and forecast the load. The next section presents an overview of load characterization and forecasting.

5. Load Characterization and Forecasting

The objective in load characterization is generally to represent in some way the load that is required to obtain a load-service relationship. That is, represent the load in enough detail to used in either a benchmarking or modeling approach to obtaining the load-service relationship.

The load on the system can be represented in some abstract way by collecting some resource or functional oriented data and reducing it to model the load. Here, resource oriented work units would be designed to consume system resources such as CPU, memory, I/O, etc. Functionally oriented work units are those that do some predefined function such as FORTRAN compile, text-edit, electronic mail, and even DB query or update. Representing the work units in this way is sometimes referred to as creating synthetic workloads. Of course, one could combine the functional and resource oriented approach in an attempt to create synthetic workloads for benchmarking.

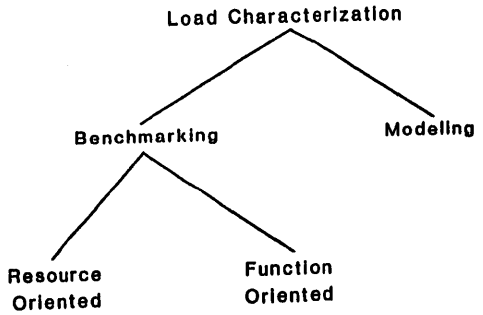


Figure 5.1

Several tools can be used to help create the synthetic workload(s) including hardware/software monitors, log files, survey data and even console logs. These tools might interrelate in the following way (see figure 5.2).

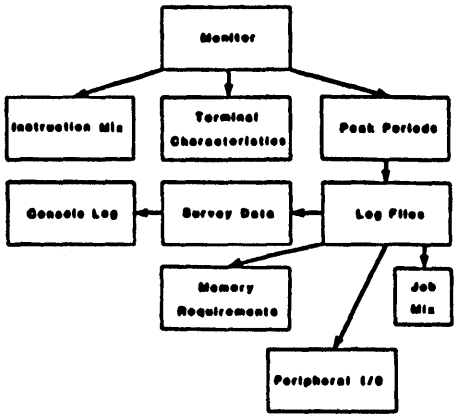


Figure 5.2

The statistical approach known as *clustering* has been used successfully to identify a small set of resources consuming work units. The reader is referred to work by Artis for more information [Artis, H. P., "Capacity Planning for MVS Computer Systems," D. Ferrari, ed., Performance of Computer Installations, North Holland Publishing, 1978]. Appendix C contains additional diagrams that will be used in the actual presentation.

6. Performance Management

Introduction

THE SYSTEM LIFE CYCLE MODEL

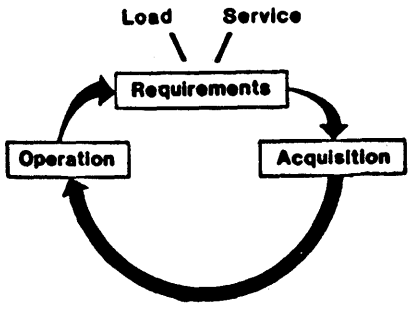


Figure 6.1

The System Life Cycle Model has previously been shown to accurately reflect the different phases of a computer system, from inception through actual modification and change. The requirements phase involves the specification of functions and perhaps even the behavior of a proposed computer system. For example, the requirements for a time sharing based application would specify the different types of work that are to be performed, e.g. inquiries about a customer, and they might also specify the expected response time for each different type of work. Let a work unit be defined as the accomplishment of some function(s) during some period of time, where a work unit is a task in a batch environment, and a work unit is a transaction in a timesharing environment. Given a list of required work units a system will do, an analyst moves onto the acquisition phase of the model.

Acquisition involves transferring the requirements into a set of hardware and software components to be obtained, which ultimately, will provide the functionality and behavior specified in the requirements.

Once the set of components are derived, an analyst goes thru a somewhat detailed phase of evaluating different alternatives (i.e. configurations) that are available, to derive that set of components that (1) is the most cost effective and (2) satisfies the initial requirements.

As a direct outgrowth of the acquisition phase, the analyst is, in essence, deriving a capacity management plan. This consists of three elements:

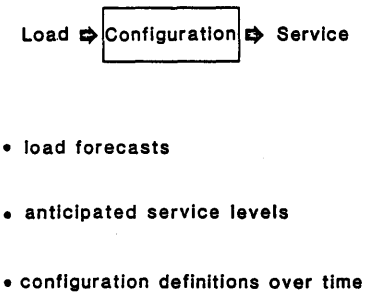


Figure 6.2

These elements address the primitive model of a computer system, namely the expected load on the system over time, the expected level of service the system provides over time, and the different configurations of both hardware and software over time that will handle the expected load.

Comparing alternatives in the acquisition phase of the system life cycle model will provide different load/service relationships for each unique configuration. Using these relationships, the analyst can define control limits for the load and service levels.

Thus, if load is increasing (or if service is degrading) at a rate faster than expected, changes to the capacity management plan should be reviewed. For example, a new cpu whose need is expected one year from the initial operation of the system, may be required after only six months, because of either a higher load than expected or because of a lower level of service than what is required.

By measuring the system regularly, the analyst can compare actual load/service levels with control limits established from the acquisition phase. Thus, a performance management plan is conceived, within the operations phase of the system life cycle model.

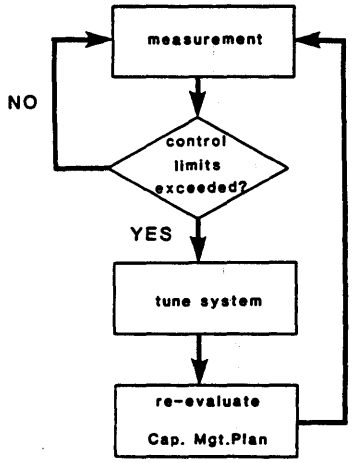
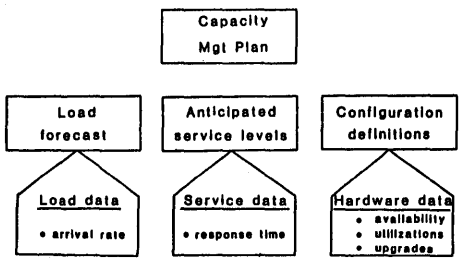


Figure 6.3

Elements of a Capacity Management Plan

Before measurement of the system, it is assumed that a capacity management plan has been defined. As stated before, it is composed of 3 elements: a load forecast, expected levels of service, and a configuration definition over time. "Levels" of load and service are defined as control limits. Here, the problem of identifying the data items that are used to compare against control limits is addressed.



Elements of a Capacity Management Plan
Figure 6.4

Load forecasts are composed of arrival rates for the different work units entering the system. For example, the number of FORTRAN compiles in a given hour, or the number of arrivals of a specific time-sharing transaction in a day. Arrival rates for each unique work unit entering the system could later be compared against control limit values for these work units.

Service levels are composed of processing or throughput rates for the different work units acted on by the system. For example, the number of FORTRAN compiles completed in a hour, or the number of completions of a specific transaction in a day. Service level data items may also address the response time of a work unit. For example, control limits for specific time-sharing transactions may have a maximum response time of ten seconds, and a minimum response time of three seconds. A natural question is posed here; namely, why worry about not achieving a minimum response time? If this is the case, one may want to delay any planned configuration changes e.g., hardware upgrades.

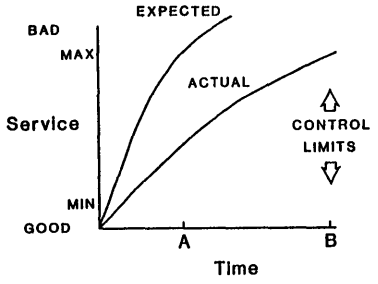


Figure 6.5

By observing that the minimal response time is being achieved later, this may imply that time A, the time at which a hardware upgrade is to be made, is too early. It may be possible to delay the upgrade until time B. All these questions are addressed in greater detail when discussing the re-evaluation of the capacity management plan.

Hardware data is composed of three components: availability information, resource utilizations, and component upgrades/replacements.

Availability information analyzes the time between component failures, and the time required to repair the failed components. Though intuitively obvious, it is worth stating that during a component repair period, little, if any, use of the system will be made. Thus, it is important in defining the times at which hardware upgrades will be made, that mean time to repair and mean time between failures be carefully considered. Vendors usually supply this information by component.

Resource utilizations are measures of how much (little) hardware components are being used. For instance, cpu 70% busy, disk drive 1 is 20% busy are examples of resource utilizations. The resources of a computer system fall into three classes: cpu, memory, and I/O. These utilizations are traditionally used to spot "bottlenecks" in system throughput, resulting in a degradation of service. Control limits for resource utilizations are usually established by "rules-of-thumbs"; that is, intelligent guesses are made based upon previous observations of similar systems using these components.

Data used in a capacity management plan can be partitioned into two classes: low versus high levels of resolution. Low resolution data items correspond to viewing load, configuration, and service with respect to the entire system. For example, low resolution load data would be number of tasks arriving to the configuration in a certain time, perhaps being further classified by type of work unit. Examples of low resolution data items are in Figure 6.6.

Low Resolution Data

Load	Configuration	Service
• # of FORTRAN compiles	• System mttr, mtbf	• 4000 transactions per day
• # of EDITs	• CPU 70% busy	• Average response time of 6 seconds

Figure 6.6

Given appropriate measurement tools and reports, the performance management process is summarized in Figure 6.7. Measurement tools like software monitors, hardware monitors, accounting files and system "snapshots" provide the necessary measurements. Graphic as well as tabular reports are necessary to analyze the status of a system. Boolean data analysis techniques have been successful in problem determination.

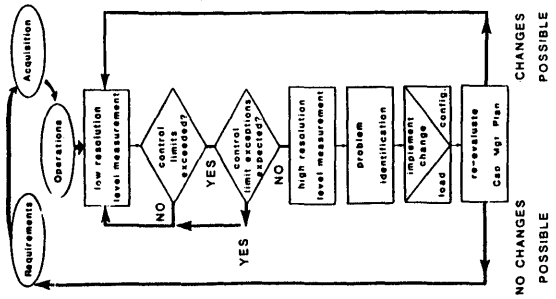


Figure 6.7

7. Summary

This tutorial introduced the reader to a variety of techniques used in the capacity planning/performance evaluation process. Where possible, references are given to direct the reader to more detailed sources of information.

8. Bibliography

It has been our experience that at the conclusion of the presentation, the audience requests additional information about many of the topics presented. Rather than providing a detailed bibliography, we have attached a list of papers by topic that we believe would be beneficial to the interested reader.

1. Orchard, R. A., "Workload, Capacity, and System Performance - A General System Definitional Distinction", Proceedings of the 28th International Conference of the Society for General Systems Research, June, 1984, New York City, Intersystems Publications.
2. Domanski, B.,"Methodologies Behind Measurement Techniques", 28th Annual Meeing of the Society of General Systems Research, Proceeding, New York City, June 1984
3. Orchard, R. A.,"A Mathematicatal Approach to the Management of Service Requirements", Proceedings, Computer Measurement Group Conference (CMG 85), Dallas, Texas, December, 1985
4. Orchard, R. A.,"A New Methodology for Computer Systems Data Gathering", 13th Meeting of CPEUG, Sept, 1977, New Orleans, LA., NBS Special Publication 500-18, US Dept of Commerce, National Bureau of Standards.
5. Orchard, R. A.,"An Algebraic Model for CPU Sizing", 18th Meeting of CPEUG, Oct, 1982, San Francisco, CA., NBS Special Publication, US Dept of Commerce, National Bureau of Standards.

Four basic quantities typify this system

T = Length of observation period

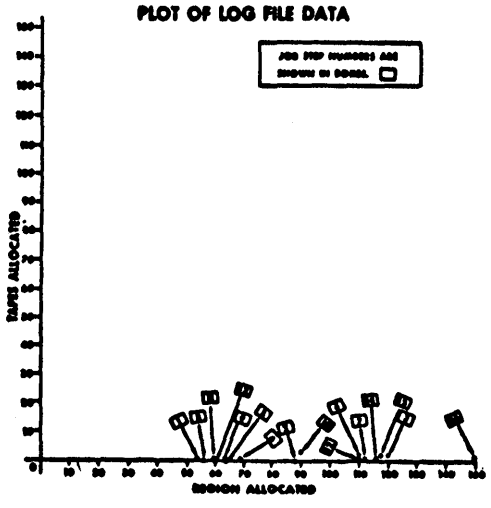
A = # of arrivals during period

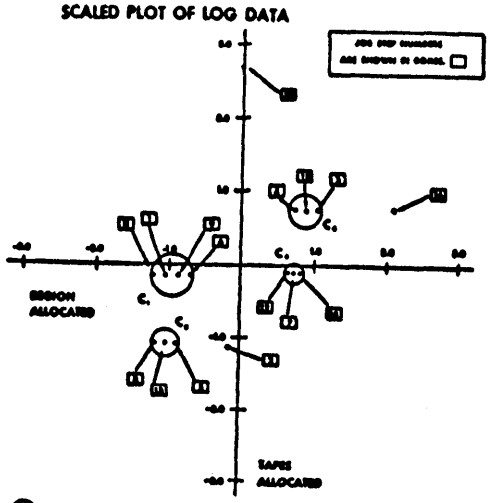
B = Total time system is busy

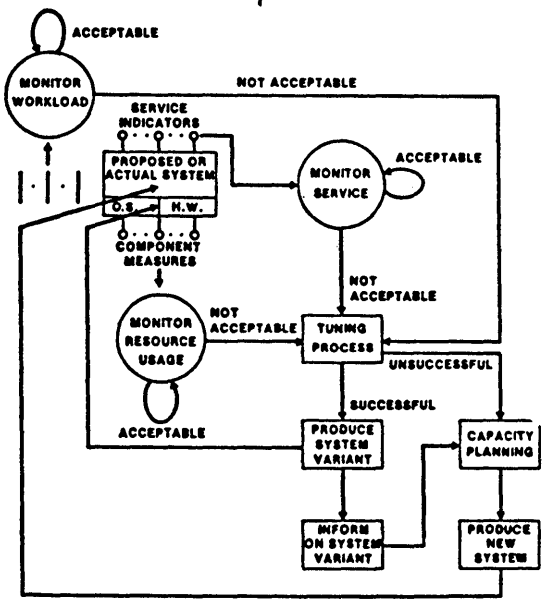
C = Number of completions occuring during period

Four important derived quantities also exist

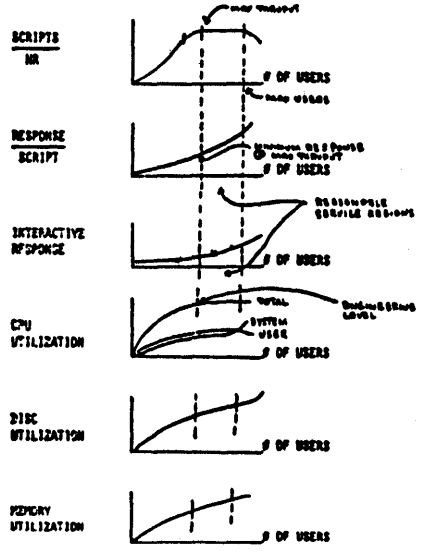
$\lambda = A/T$	Arrival rate
$X = C/T$	Output rate
$U = B/T$	Utilization
$S = B/C$	Mean service time







S = F(L,C)	PERFORMANCE PROBLEMS
C = G(L,S)	CONFIGURATION PLANNING
L = H(C,S)	CAPACITY EXHAUST



Fundamentals of Statistical Workload Forecasting

H. Pat Artis
Performance Associates
Vienna, VA 22180

Abstract: This tutorial will examine a variety of statistical methods that can be used to forecast computer workloads. The tutorial will discuss simple regression, multiple regression, and time series analysis techniques as well as statistical metrics for evaluating the goodness-of-fit of the proposed models.

1.0 Case 1: Time Based Regression

Perhaps the commonly used forecasting tool by performance analysts is linear regression. Linear regression develops a model based on the assumption that a dependent variable, *Y*, is related to an independent variable, *X*, as a linear function of a constant, *b*, and a slope variable, *m*. Time based linear regression is the most common application of this technique. To illustrate the strengths and pitfalls of the approach, a time based regression model of weekly average CPU utilization will be developed. This exercise will be used to introduce:

- residual values,
- R-squared, the coefficient of determination,
- F and t values,
- estimates of the standard error, *SE* and mean squared error, *MSE*,
- and confidence limits and prediction intervals.

One of the limitations of linear forecasting will be illustrated by introducing an example of a *step function decrease* that occurs when a processor is upgraded.

2.0 Case 2: Using Dummy Variables

The second case study will demonstrate how dummy variables can be used to address the problems of step function increases (decreases) in the dependent variable. Dummy variables, column vectors comprised of 0's and 1's, allow multiple regression models to treat one or more step functions in the historical data series as constants that are added or subtracted from the Y observations by the model prior to the development of the linear model.

While dummy variables allow multiple regression models to address step function changes in historical data series, they cannot address the problems introduced by apparently random or periodic variations in the historical data. Approaches to these problems will be discussed in the following sections.

3.0 Case 3: Multiple Regression Applications

When a single application dominates a system, or an interval during the day, the underlying business activities that drive the application often cause significant variations in the day-to-day utilizations recorded for the system. To address such environments, the analyst must work with the application's users to identify the business elements that drive the application's consumption of resources. **If the user can forecast these business elements,** then the analyst can employ stepwise multiple regression to develop models that forecast future resource requirements based on the user's estimate of the business elements. **It should be noted that the analyst can never assure that business element forecasting will work. Even if a historical relationship exists between the business elements and resource utilization, there is no guarantee that the users can accurately predict the business elements in the future.**

The case study will introduce an online order entry system with significant weekly and monthly variations in resource utilization. After the model has been developed, the use of **accounting rate table elements** to translate the predicted resource requirements into future budget dollars will be discussed.

4.0 Case 4: Time Series Applications

The final case study will introduce the use of time series analysis to address workloads that have repeating periodic variations in their resource demands. Simply described, time series analysis techniques repeatedly subtracts various periodic (i.e., *seasonal*) functions from the prior observations until the the remaining historical series of observations can be modeled by a linear regression model that represents the long term growth rate of the application. Forecasts are developed by first predicting the long term trend and then adding the periodic functions back into the predicted values.

5.0 Remarks

While statistical workload forecasting techniques provide the analyst with powerful tools for estimating future demands, they can never replace a detailed understanding by the analyst of the users' requirements and future business plans. Rather, these tools should viewed as a means of better understanding problems... not providing a panacea for their solution!

A CAPACITY PLANNING/QUEUEING THEORY PRIMER OR HOW FAR CAN YOU GO ON THE BACK/OF/AN/ENVELOPE?

Ethan D. Bolker
BGS Systems, Inc.

ABSTRACT

Most of the benefits of capacity planning for computer systems come from applying common sense principles to easily understood (if not easily measured) data. If you can estimate business growth you need only simple arithmetic to figure out how busy your computer will be. But sometimes the fact that the system is subtle does matter: it's hard to predict response times. Then you must use mathematics (instead of arithmetic) or software which knows some - typically, a package incorporating a queueing network model of your computer system. You can (and should) think of the package as a black box whose sophisticated mathematical contents need not concern you. In this top down view such products are worth what they cost because they're cost effective. But from time to time you may want to peek into the box - to satisfy a healthy curiosity, or to convince yourself that the mathematics there makes sense, so that you can use its output with confidence or explain its function to your manager. (But exercise care - you can sell your manager your services only by showing why s/he needs them, not by describing the mathematical magic which helps you do your job.)

This introductory tutorial moves from simple capacity planning arithmetic to a glimpse of queueing theory - how it solves relevant problems, and why it's for a less technical audience too - managers, secretaries, planners' husbands and wives - anyone who wants to know a little about how a queueing theory package earns its keep. It's essentially what I tell my friends (outside the industry) when they ask what I do.

1. THE NUMBERS GAME

Imagine that you are managing a bank's Cash Machines, and that you know something about how much business a particular machine will be called upon to do. To be specific, suppose you anticipate that

30 customers will visit the Machine in an hour.

Each will use it for a minute and a half.

You need answers to two questions

How busy is the Cash Machine?

How long do customers wait?

Before we answer them, we can note a pattern. We know about customer behavior: how often they arrive (30 customer/hour) and how much service each requires (1.5 minutes/customer). What we need to know depends on how the customers interact with the system: the percentage of time the Machine is busy and the average customer wait.

You can measure all of these numbers by hiring a High School student to watch the Machine, count customers and time their transactions. Or you can program the Cash Machine to keep track of the first three while it does the business it was bought to do, and program the door to the room in which the Machine lives to find the last one. That's "Computer Measurement". But CMG members don't just measure, they plan. In a planning scenario the first two numbers - the customer arrival rate and the service each customer requires - aren't measured, they're part of the business forecast. The planner needs to predict the second two: how busy the machine will be, and the waiting times. That's where mathematics come in.

Simple arithmetic tells us about the Machine:

$$30\ \frac{\text{customers}}{\text{hour}} * 1.5\ \frac{\text{minutes}}{\text{customer}} = 45\ \frac{\text{minutes}}{\text{hour}}$$

The Cash Machine is in use for 45 minutes each hour, so it is busy 75% of the time. The underlying algebra is quite straightforward, and not all mysterious. It even has a name - the Utilization Law:

arrival rate * service time = utilization

(when the named quantities are expressed in appropriate units).

2. THE WAITING GAME

The second question is harder. Let's reflect on it a bit. Suppose the 30 hourly customers came at scheduled times 2 minutes apart, and that each customer was an average customer and did just 1.5 minutes of banking. Then no one would have to wait, and the Cash Machine would rest for 30 seconds between jobs. But common sense and our own banking experience tell us these are unrealistic assumptions. Customers come one every two minutes on the average. But sometimes there will be long idle times; sometimes several customers will arrive nearly at the same time. And although on the average customers spend 1.5 minutes at the Cash Machine some are just getting FastCash while others do a week's worth of banking all at once. The more variability of this kind the more time customers will spend waiting while someone else is using the Cash Machine - even though the average arrival rate and average service time stay the same. Predicting the response time - the average time spent (waiting and working) is harder than calculating the utilization of 0.75. The formula which helps us out is

$$\text{response time} = \frac{\text{service time}}{1 - \text{utilization}}$$

In our example, since the Cash Machine is 75% busy, and the service time is 1.5 minutes,

response time = 1.5/(1-0.75) = 6 minutes

The formula we've just used is not nearly as straightforward as the Utilization Law. To discover it requires some mathematics at about the college (Calculus) level. Fortunately, we needn't discover it - we need only understand it. And to see why it's reasonable all that's required is some High School algebra, which we'll see in the tutorial, and Little's Law, which says that

$$\text{arrival rate} * \text{response time} = \text{Average number of customers at the Cash Machine}$$

Don't worry. This is not an infinite regress. In the tutorial we'll see where Little's Law comes from, and so assemble all the pieces.

To use Little's Law in our example we must make the units match. The service and response times are measured in minutes. The arrival rate, 30 customer/hour, is 0.5 customers/minute. Then the average number of customers at the Cash Machine is 0.5 * 6 = 3.

The response time calculation takes some getting used to. People often find its predictions counter intuitive. For example, even though our Cash Machine is idle 25% of the time, so that 1/4 of the arriving customers need not wait at all, on the average a new arrival will see 3 people already at the Machine (including the one currently using it), and will use 6 minutes of his time to do 1.5 minutes of banking business. He waits 4.5 minutes, for the 3 customers ahead of him in line.

The answers to our two questions sharply focus the capacity planning dilemma you face as the Cash Machine Manager: dissatisfaction both because

the 75% utilization is too low.

the 4.5 minute wait is too long.

3. WHAT IF?

The response time calculation vividly quantifies the compromise you must make. It shows you exactly how decreasing your unhappiness with one of the answers will increase it with the other.

For example, if you expect Cash Machine customers will soon be arriving at a rate of 36 rather than 30 per hour (a 20% increase) then Machine utilization will increase to 90% (which makes you happy) but the response time will increase to 15 minutes, of which 13.5 is spent waiting. This is three times the current 4.5 minute wait, which makes you - and the customer - unhappy.

So far all the arithmetic fits on the back of an envelope.

You know your customers will not tolerate those long waits (they can't appreciate the fact that you like to keep the machine busy) so you decide to install a second machine next to the first.

What are the answers now to the two central capacity planning questions? Will each machine be busy 45% of the time? Will the average wait be half of 13.5 minutes? Again, the first question is easy; the second is hard.

Each of the two Machines will indeed be in use just 45% of the time. That follows from the Utilization Law. But the average number of customers present at the pair of Machines turns out to be a mere 1.3, and, on the average, an arriving customer will wait less than two minutes before his turn comes. We will briefly discuss the mathematics which produces these numbers. The resulting formulas are still simple enough to evaluate with a calculator, but showing where the formulas come from is a little too complicated for an introductory tutorial.

4. PARALLEL PROCESSING

Moreover, if we install the second Cash Machine and measure waiting times we will discover that our predictions are too optimistic (and that's the wrong direction in which a planner should err.) To understand why we must think more carefully about what happens when both Cash Machines are in use.

Let us analyze the 1.5 minutes each customer spends standing at the lone Cash Machine before we install the second one. Some of that time, say, 1 minute, is time spent reading the instructions on the screen, deciding what to do, removing cash or filling out the envelope to hold the deposit. The remaining 0.5 minute is the time it takes the computer behind the Cash Machine to process the transactions.

To study that half minute, first imagine the bank in the days before computers, when the Cash Machines were tellers. The teller took the customer's request to the bookkeeper, (suppose there was just one bookkeeper) who asked a clerk to bring him the customer's account record from the appropriate filing cabinet. When the record arrived the bookkeeper did the necessary arithmetic, sent the teller back with an answer for the customer and sent the clerk back to file the updated account record. This sequence of jobs, which used to take, say, 3 minutes, accounts for the 30 seconds we measure now when computers do the bookkeeping and record management.

Look how wasteful it was (and is). The bookkeeper was (presumably) better paid than the clerk and the teller, but was idle most of the time, since she did her arithmetic in a small fraction of the 3 minutes of processing time. The retrieval and refiling of the record took up the rest. That's why it would be economical to hire several tellers and clerks to keep one bookkeeper busier.

But then each customer would spend more than 3 minutes at the teller's window. The time spent actually working on his business would be the same, but customers' jobs could get in each others' way. Even though several tellers could serve several customers simultaneously the bookkeeper worked on only one record at a time. And only one clerk could use a filing cabinet at a time. (Although two clerks could get accounts from separate filing cabinets while the bookkeeper worked on a third account.) Therefore, some of the time one customer's work would have to wait while the bookkeeper was busy with another's, or while the right filing cabinet was in use. These internal delays account for the time each customer waits in excess of the 3 minutes of real processing his job requires. Moreover, our analogy works even when the teller is a Cash Machine and the bookkeeper and file cabinets have been replaced by a computer. The computer can process only one job at a time at the CPU (the Central Processing Unit, which is the computer's bookkeeping hardware) and can look up only one account at a time on each disk (the computer equivalent of a file cabinet), although it can look up one account while it is processing work for another. Therefore, as before, some of the time one customer's work will have to wait while another's is being done. Similarly, when there are two Cash Machines active simultaneously the computer needs more than 1/2 minute to process each customer's work. Just how much more is the crucial question.

5. BLACK BOXES

We will conclude the tutorial by discussing how one might calculate how long each job will wait at the disks and at the CPU and thus predict how long the average customer will have to spend between the time he arrives to wait for the Cash Machine and the time his banking is complete. The calculations resemble the ones we have just looked at but they are so much more complicated that they can be done rapidly and economically only by a computer program - a software package which knows queueing theory. As a planner, you speak to the package in the language you use to describe your computer system - not the mathematical language it uses internally to predict how the system will behave. It answers you in your terms too, telling you about utilizations and response times. You use those numbers to plan computer capacity to cope with expected business needs. And now, when you do, you'll know a little more about where those numbers come from.

REFERENCES

The literature on the applications of queueing theory to computer performance evaluation is extensive. Here are a few suggestions for further reading.

[1] Arnold O. Allen, Probability, Statistics, and Queueing Theory with Computer Science Applications, Academic Press, 1978.

[2] Jeffrey P. Buzen, Fundamental Operational Laws of Computer System Performance, Acta Informatica 7,2 (1976), 167-182.

[3] Jeffrey P. Buzen and Peter J. Denning, The Operational Analysis of Queueing Network Models, Computing Surveys 10,3 (September 1978).

[4] Edward D. Lazowska, John Zahorjan, G. Scott Graham, Kenneth C. Sevcik, Quantitative System Performance, Computer System Analysis using Queueing Network Models. Prentice Hall, 1984.

HIGH-PERFORMANCE PROCESSOR DESIGN

Bill McCormack, Jay Michlin, Larry Brisson, and Andy Martin
Amdahl Corporation

ABSTRACT

One purpose of large mainframe computers is to execute machine instructions in the shortest time possible. In this paper, we discuss some of the factors introduced into processors that help achieve this purpose. In particular, we explain how computer designers use parallelism, caches, and pipelines to make modern mainframe computers faster. We also discuss some of the factors that limit the effectiveness of these features.

INTRODUCTION

Processor design involves translating a high-level view of the design into a detailed description of the implementation. In this process, designers have some flexibility and must make many decisions that have reliability and performance implications. In this tutorial, we discuss some of the design decisions and the impact they have on processor performance.

In the next section, we describe the performance metrics we use in this paper. We also discuss some of the reasons for using these metrics. Following that, we discuss the effect that caches, parallelism within the execution of a single instruction, and processor pipelines have on performance. The use of these and other design features greatly improves the performance of a processor.

To show the performance effects, we break down the execution of a typical instruction into the steps that must be performed and the approximate number of processor cycles required for each step. We then show the reduction in processor cycles used as a result of introducing caches, parallelism, and a pipeline. Finally, we discuss some of the factors that limit the effectiveness of these features.

MEASURES OF PERFORMANCE

Many articles have been written debating the merits of various performance measures. Two measures frequently discussed are internal and external throughput rates [1]. Without getting into that discussion, we can say that both measures are valid when used in context, and both are affected by factors other than just the processor. For example, real storage, I/O bottlenecks, and the number of processors all affect these throughput rates.

When designing the processor itself, one must focus on a measure that reflects only the processor. Designers commonly use one of the following two measures: mean instruction execution rate (IER, usually stated in millions of instructions per second or MIPS) and average number of cycles per instruction. We discuss both of these and their relationship.

IER is widely used (and abused) as a measure of performance. As discussed by MacDougall [2], the execution rate of applications can vary greatly. In addition, changes to the architecture (e.g., using microcode to replace functions previously performed by the operating system) or changes to the code in the operating system can cause the mean instruction execution rate to decrease, but the system throughput to increase [1]. Thus, in most customer environments, instruction execution rate is an inappropriate performance measure. However, IER is a useful measure for comparing different processors or processor designs of the same architecture.

MIPS is the number of instructions executed per second; thus, if we take the reciprocal, we get the average time per machine instruction in microseconds or nanoseconds per instruction. (LaMaire and White [3] use nanoseconds per instruction in their study.)

Average time per instruction can be further broken down into its two components: the average number of machine cycles per instruction (I) and the cycle time of the machine (C) in microseconds. Thus, we can write

$$MIPS = 1 / (C * I).$$

Processor designers usually treat the two factors, C and I, separately. A cycle is the basic unit of time for the processor. The technology available largely determines the cycle time; current cycle times for large mainframes range from 10-20 nanoseconds. One reason for treating the two factors separately is that the cycle time of the processor may not be finalized until late in the design of the machine. Thus, the designers work with a cycle time goal.

Some instructions can be completed in one cycle, but many cannot. The number of cycles required for each instruction depends on the instruction, its operands, "nearby" instructions, and the design of the processor. Thus, the cleverness of designers is reflected in the number of cycles per instruction.

Just like IER, the average number of cycles per instruction can be abused. If only very simple instructions are executed, then it is possible to have a low number of cycles per instruction (and hence a high IER). The average number of cycles per instruction also varies from application to application and environment to environment. Thus, for the same machine, one often produces different estimates for the average number of cycles depending on the environment. These estimates are based on the types and frequencies of the instructions found in each environment. MacDougall [2] discusses a methodology for these estimates.

In the remainder of this paper, we concentrate on the number of cycles per instruction. However, for our examples we assume a cycle time of 15 nanoseconds (0.015 microseconds); this assumption allows us to translate the cycles per instruction into MIPS.

TYPICAL INSTRUCTION EXECUTION

In Figure 1, we show the steps required to execute a representative machine instruction. Beside each step, we list the approximate number of machine cycles needed for that step.

	Cycles
Fetch Instruction from Memory	20
Decode Instruction	1
Generate Address of Storage Operand	1
Translate Address	1
Fetch Operand from Memory	20
Read Register Operand	1
Perform Arithmetic/Logical Operation	1
Write Result (Usually to a Register)	1
Compute Next Instruction Address	1
Total:	47

Figure 1: Typical Instruction Execution

First, we must fetch the instruction from the physical memory of the processor. The memory cycle time (time to get something from memory) is typically much longer than the (processor) cycle time; here we assume it takes 20 cycles. Then, we must decode the instruction to determine what is to be done (e.g., an ADD, a BRANCH). The next three steps are used to fetch the operand from memory. First, we generate the virtual address of the operand. Then, we must translate the virtual address into a physical address (here, we assume a system with virtual addresses). Then, we can fetch the operand from memory.

We assume that the second operand is in a register; thus, no memory access is required for this operand. Finally, we are ready to perform the operation itself. In this case, we assume a simple operation that can take place in one processor cycle. Examples of such operations might be an integer add, a load, or a store. Many instructions take more than one cycle; examples include floating point arithmetic operations and moving many bytes of data.

After execution, we assume the result is stored into a register. Lastly, we determine the address of the next instruction to be executed so we can repeat these steps.

In total, it took 47 cycles to complete a simple instruction involving a one-cycle operation. To translate this into MIPS, we apply the formula stated earlier: MIPS = 1 / (C * I), where C = 0.015 microseconds and I = 47. Thus, this processor executes at about 1.4 MIPS! Current large mainframe processors are typically much faster than 1.4 MIPS. In the next few sections, we discuss some ways to reduce the average number of cycles per instruction.

CACHE STORAGE

In our example, over 80% of the cycles were required to fetch the instruction and operand from memory. Thus, the first way to reduce the average number of cycles is to reduce the cycles spent accessing memory. Commonly, a cache or buffer is used so that very few processor cycles are needed to access something from storage. (The terms cache and buffer are both used widely; in this paper we use the term cache.)

Cache is high-speed storage located between the CPU and main storage. Cache speeds up memory accesses but is transparent to the programmer and operating system. In most multiple processor systems, each processor has its own cache. The technology for the cache is much faster and more costly than that used for the processor memory. Smith [4] discusses caches in detail.

The cache size of current processors is less than 100 kilobytes or about one-thousandth the size of the main memory on processors. Cache sizes and implementations differ among manufacturers and even among processors. For example, each processor of an Amdahl 5890 [5] has two caches: a 32-kilobyte cache for instructions and a 64-kilobyte cache for operands. Each processor in an IBM 3090 [6] has a cache of 64 kilobytes that is used for both instructions and operands.

Typically, in supervisor state, we can expect about 19 out of 20 memory requests to be found in the cache; thus, a "cache miss" occurs only 1 time out of 20. For problem state, cache misses are even less frequent. If we assume the two memory requests in Figure 1 are found in the cache and a cache access takes only one cycle, then we reduce the number of cycles required for the instruction from 47 to 9. Translating this to MIPS, we have improved the execution rate of the processor from 1.4 to 7.4 MIPS.

PARALLEL OPERATIONS

By adding a cache, we improve the processor's performance by a factor of about five, but we still have a processor that is slow by today's standards. The next design improvement is to execute some of the steps in parallel. The steps we are describing take place in each processor. Figure 2 shows the same steps as Figure 1, only we have arranged them in a diagram that shows the actual order in which the steps must be performed. Figure 2 is similar to a Program Evaluation and Review Technique (PERT) diagram often used in planning a project.

The first two steps must be completed in order: fetch the instruction and decode it to determine what is to be done. Once we have decoded the instruction, we can generate the address of the operand and at the same time prepare to read the register operand. After the operand address is generated, we can perform a number of steps in parallel.

First, the address is a virtual address; thus, we can begin translating the virtual address into a real address. We can also generate the address of the next instruction. Figure 2 shows the storage operand being fetched while the translation is in progress. These two steps can take place concurrently because we assume the cache is accessed using the virtual address.

After the operand fetch is completed, the arithmetic or logical operation is performed. When the operation is completed, the result is written into the register. We have already generated the next address, so the process can begin over immediately. Before determining the improvement that this parallelism achieves, we should discuss the address translation step in more detail.

Generally, when we use the virtual address to access the cache, we fetch the correct bytes. The address translation (or address validation) step goes on in parallel and determines whether or not we accessed the correct cache entry. If the correct bytes were fetched, then execution can begin; otherwise, the proper operand is fetched. The address translation is invisible to the program and the operating system, and, typically, the cycle needed for address translation is eliminated.

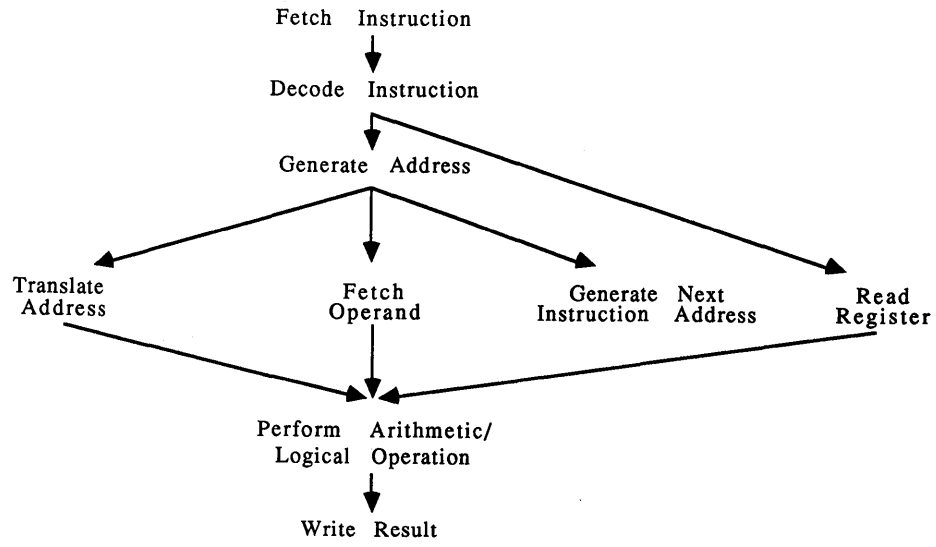


Figure 2: Parallel Operation of Instruction Execution

Thus, by using some parallelism, we effectively eliminate three cycles: the cycles needed to translate the address (1 cycle), read the register operand (1), and compute the address of the next instruction (1). The number of cycles per instruction is down to 6; thus, the execution rate is now about 11.1 MIPS.

INSTRUCTION PIPELINES

The final step in our process is to add another form of parallelism: a pipeline. We should think of a pipeline as an assembly line; for example, a traditional automobile assembly line. As the frame of the car moves along the line, different steps are performed until the car reaches the end. At any point in time, many cars are being worked on; but each is in a different stage of completion. As long as everything functions properly, completed cars should come off the assembly line regularly.

A processor instruction pipeline is very similar. An instruction moves through the pipeline, and at each stage the necessary steps are performed. After every cycle, each instruction moves to the next stage. Thus, if nothing goes wrong, one instruction will complete every cycle. Pipelines are used in large mainframes such as the Amdahl 5890 and IBM 3090 series [5,6]. Again, each processor has its own pipeline.

Figure 3 shows a pipeline with six stages. The six stages correspond to the six crucial phases in Figure 2. Each phase in Figure 2 takes only one cycle; thus, after every cycle, every stage is complete, and all instructions in the pipeline move forward. To explain Figure 3, we can describe the progress of the first few instructions as follows: At the end of cycle 1, instruction 1 has been fetched. At the end of cycle 2, instruction 1 has been decoded and instruction 2 has been fetched. At the end of cycle 5, instruction 1's operation has completed, instruction 2 has its operand, and instruction 5 has just been fetched.

Beginning with cycle 6, an instruction completes every cycle: instruction 1 finishes after cycle 6, instruction 2 after cycle 7, etc. Once the pipeline is "full," the processor can finish one instruction every cycle; thus, the average number of cycles per instruction is just 1. Translating to MIPS, we have a single processor capable of executing almost 67 MIPS!!

PIPELINE DELAYS

Obviously, processors don't achieve 67 MIPS consistently; we discuss some of the reasons in this section. To achieve 67 MIPS (one cycle per instruction) means no pipeline delays can occur; that is, every stage in the pipeline takes only one cycle and nothing "goes wrong."

	Pipeline Stages					
Time	Fetch Instruction	Decode Instruction	Generate Address	Fetch Operand	Perform Operation	Write Result
1	1					
2	2	1				
3	3	2	1			
4	4	3	2	1		
5	5	4	3	2	1	
6	6	5	4	3	2	1
7	7	6	5	4	3	2
8	8	7	6	5	4	3

Figure 3: Instruction Pipelines

The simplest pipeline delay occurs when some instruction cannot be completed in one cycle (e.g., moves, divides). When a delay occurs (e.g., in the *perform operation* step), all instructions following that operation must wait. Again, the analogy to an automobile assembly line is useful: all other stages (workers) are idle until this stage can be completed and the pipeline restarted.

Figure 4 shows the same pipeline as Figure 3; however, now we assume that the execution of instruction 2 takes three cycles. For two cycles, no instructions can move in the pipeline. The effect of this delay is seen at Times 7 and 8, when no instructions finish.

A number of other pipeline delays can also occur. In our example, we assumed that all memory accesses were found in the cache and fetched in one cycle. We also assumed that the virtual address used to access the cache for the storage operand always accessed the right cache entry. If any of these were not true, delays would occur.

The final delay we wish to discuss is commonly called a pipeline interlock. Basically, we are referring to delays in the pipeline caused by other instructions in the instruction stream. Interlocks depend largely on the actual sequence of instructions executed, and not just on the frequency of the instructions. An example of a code sequence causing an interlock follows:

```
A  R1, X
L  R2, 0 (R1)
```

The first instruction adds the contents of location X to Register 1 and writes the result in Register 1; the second instruction uses the contents of Register 1 to determine the address to be loaded in Register 2. MacDougall [2] calls this type of interlock a "generate interlock."

The key is that the second instruction is unable to generate its (operand) address until the first instruction completes. Figure 5 shows the pipeline in this situation. After decoding the second instruction, the instruction cannot proceed to generate the address until the ADD operation is completed. Thus, delays (or holes) appear in the pipeline, and no instruction completes in some subsequent cycles.

Figure 5 actually shows more than just the delays; it also shows a common trick used to reduce the amount of delay. The address generation step begins after the *perform operation* is completed, but before the result is written to Register 1. The processor can recognize that a subsequent step in the pipeline needs the result of the *perform operation* step. Thus, when the operation is completed, a special data path sends the result to the needed step as well as to the result register. This procedure eliminates one cycle from the delay. This type of trick is commonly called a "bypass."

In the design of large processors, saving cycles using pipeline bypasses can greatly affect the performance of a processor. Numerous interlocks have been identified; MacDougall [2] discusses many interlocks and some of the bypasses used to eliminate them or reduce their effects.

	Pipeline Stages					
Time	Fetch Instruction	Decode Instruction	Generate Address	Fetch Operand	Perform Operation	Write Result
1	1					
2	2	1				
3	3	2	1			
4	4	3	2	1		
5	5	4	3	2	1	
6	6	5	4	3	2	1
7					2	
8					2	
9	7	6	5	4	3	2
10	8	7	6	5	4	3

Figure 4: Pipeline Delays - Long Execution Time

	Pipeline Stages					
Time	Fetch Instruction	Decode Instruction	Generate Address	Fetch Operand	Perform Operation	Write Result
1	1					
2	2	1				
3	3	2	1			
4			Delay	1		
5			Delay		1	
6	4	3	2			1
7	5	4	3	2		
8	6	5	4	3	2	
9	7	6	5	4	3	2
10	8	7	6	5	4	3

Figure 5: Pipeline Interlocks

CONCLUSIONS

We began our discussion with a hypothetical processor requiring 47 cycles per instruction, and then, through pipelines, caches, and parallelism, we have shown that it is possible to complete one instruction every cycle. However, processors cannot average one cycle per instruction because of various cache and pipeline delays. Generally for 370 architecture processors, the average value is between three and four cycles. Again, the actual value (that is, the effectiveness of the cache, pipeline, and parallelism) depends on the environment and the application. This fact is clearly shown in LaMaire and White [3], who describe the fluctuation in nanoseconds per instruction (cycles per instruction multiplied by the cycle time) for a production workload over the course of a day.

REFERENCES

[1] Radecki, K.L., "Introduction to Large Processor Capacity and Performance Evaluation," IBM Washington Systems Center Technical Bulletin GG66-0232-00 (1986).

[2] MacDougall, M.H., "Instruction-Level Program and Processor Modeling," IEEE Computer, pp. 14-24 (1984).

[3] LaMaire, O.R. and W.W. White, "Processor-Level Workload Characterization," IBM Research Report RC 11385 (1985). (Also presented at the International Workshop on Workload Characterization of Computer Systems and Computer Networks, Pavia, Italy, October 1985.)

[4] Smith, A.J., "Cache Memories," ACM Computing Surveys, Volume 14, Number 3, pp. 473-530 (1982).

[5] Amdahl Corp., 5890 Functional Characteristics, Amdahl Publication Number MM119388-002 (1987).

[6] Tucker, S.G., "The IBM 3090 System: An Overview," IBM Systems Journal, Volume 25, Number 1, pp. 4-19 (1986).

IMPLEMENTING NETWORK CAPACITY PLANNING

Toby Olberding
US West Advanced Technologies

ABSTRACT

Communication network complexity and size are increasing at a faster rate than any other segment of data processing. During this time of growth, while the per unit cost of the hardware has dropped, the per unit cost of the network has increased. The visibility into the network has also decreased, while the demands have increased.

Corporate plans are carried out with the assumption that the network will successfully provide the service needed to meet the business demand. A network that operates in a degraded mode, or fails completely, causes repercussions throughout the corporation. Degraded network service translates directly into lost productivity and lost business. An unhealthy network means customer service-level objectives are not met, tasks are not competed on time, budgets are overrun, and corporate business plans fail.

This paper is a discussion of how to get started in network measurement and planning. A secondary purpose is to provide a sample project plan to implement the measurement and planning. The topics covered are Workload Characterization, Workload Forecasting, Service-Level Objectives, Integrated Capacity Planning, and Management Reporting.

1.0 INTRODUCTION

One of the success factors for a Management Information Systems (MIS) organization is the proper planning for the MIS data network. To accomplish the planning for the network the following project plan should be followed.

There is need for an increased awareness of what is happening on the network. The need requires that measurement of the network be added to our system measurements to allow us a window into the network.

The MIS organization has an immediate need to strengthen its visibility into the network. This visibility will be used for improved services to the users and provide more realistic service level agreements.

Communication network complexity and size are increasing at a faster rate than any other segment of Data Processing. During this time of growth, while the per unit cost of the hardware has dropped, the per unit cost of the network has increased. The visibility into the network has also decreased There is need for an increased awareness of what is happening on the network. The need requires that additional tools be added to our system to allow us a window into the network.

The expected benefits for the project are:

1. Increased customer service
2. Reduced over expenditures on Network services and hardware
3. Monitoring of Network Workload

1.1 OVERVIEW OF PROJECT

The following is a project plan to carry out Network Capacity Planning for a MIS organization. The need for capacity planning is real if not obvious. We at MIS need to know how the network is growing, who is using the network, and what the future might bring. To answer the questions; the solution is a multiphase approach to Network Capacity Planning.

The phases are:

- Phase 1 -- Workload Characterization
- Phase 2 -- Workload Forecasting
- Phase 3 -- Service-Level Objectives
- Phase 4 -- Integrated Capacity Planning
- Phase 5 -- Management Reporting

The phases are not serial; but have some concurrency.

The need for the network information is important to the success of the MIS organization and the corporation.

1.2 TIME TABLE

The following figure shows the time table for implementing Network Capacity Planning.

Management Approval	Start
Host Software Order	+ 2 Days
Software Delivery	+ 3 Weeks
Software Installation	+ 6 Weeks
Data Collection Start	+ 6 Weeks
Workload Characterization Report	+ 2 Months
Initial Workload Forecast	+ 2 Months
Workload Forecast with Trending	+ 4 Months
Service-Level Objectives	+ 3 Months
Integrated Capacity Planning	+ 6 Months
Management Reporting	+ 2 Months

Figure 1. Network Capacity Planning Time Table

2.0 PHASE 1 -- WORKLOAD CHARACTERIZATION

The first phase of the project will be to collect the information about who is using the network and how they are using the network.

2.1 OVERVIEW

The workload characterization is a process to find out who is using the network, how much is the network used, and for what purpose. The first step in network workload characterization is to gather data about the use of the network. The data can be gathered by using several different types of data collection tools.

The basic concept of workload characterization is that the study of the network can be simplified by the identification of a number of similar "resource patterns," which can be used to statistically represent a network's workload.

Workload characterization is the development of an understanding and a description of today's workload and the service provided. How a workload is characterized depends on the intended use of the characterization. Will the workload characterization be used for benchmarking, management reporting, system modeling, or forecasting?

It is essential that the eventual use of the workload characterization be known before any significant characterization begins.

2.2 SOURCE OF INFORMATION

For this paper the workload characterization will be used for system modeling, management reporting, and forecasting.

The source of the information is the inputs and outputs to the network from the host system. The information is gathered from software monitors that reside in the host and front-end processors.

Software based tools offer the best way to obtain the required information.

2.2.1 REQUIRED MEASUREMENT TOOLS

The required measurement tools are ones that measure the total network constantly. Total network measurements include measurements of all the lines, nodes, and terminals that are attached to the network. The network measurements should be active during the period that the network is active.

Usually software measurement tools provide the best solution by providing consistent measurements.

2.3 EXPECTED RESULTS

The expected results of the data collection will:
characteristic the network workload; determine who is using the network; how much of the network is being used; and what service-levels are being provided to the user community.

The deliverable will be a monthly report detailing how the network is being used. The information that will be provided is shown in the following figure.

1. Availability by: a. Line b. Node 2. Response time by: a. Company b. Line c. Node 3. Usage by: a. Company b. Line c. Node

Figure 2. Phase 1 Deliverable

2.4 TIMETABLE

The following figure shows the time table for implementing the workload characterization phase of the network capacity planning effort.

Management Approval	Start
Host Software Order	+ 2 Days
Software Delivery	+ 3 Weeks
Software Installation	+ 6 Weeks
Data Collection Start	+ 6 Weeks
Workload Characterization Report	+ 2 Months

Figure 3. Phase 1 Time Table

3.0 PHASE 2 -- WORKLOAD REQUIREMENTS FORECASTING

A key part of network capacity planning is to obtain the workload forecast and requirements for users of the network.

3.1 OVERVIEW

Workload forecasting is used to estimate the resources requirements of an application at a future point in time. The estimation techniques range in complexity from simple growth rates evaluation to complex statistical analysis.

The two basic components of workload forecasting are to forecast growth of current applications, and to forecast new applications. Clearly, the planning analyst requires input from users and management to do an acceptable job of forecasting. Without such input the best that can be done is to extrapolate the current applications in a statistical manner based on historical data For new applications the best that can be done is to include a contingency factor in the planning process, which may allow unknown new applications to be acceptably processed.

The management of the network needs to remember the growth of the network. To accomplish this awareness of growth, a capacity planning function needs to be included in network management.

3.2 SOURCE OF INFORMATION

Workload forecasting, like any other planning, must start at the top. In this case, the top means the users' business plan. The business plan must provide the guidelines for preparing the data processing workload forecast.

A recommendation is that users first forecast changes in their applications as an increase or a decrease for services that MIS provides. Then based on business plans, in business-oriented units familiar to the users, the planning process can start. The planning process must then transform these business units, often called "Natural Forecast Units" or "Key Volume Indicators", into input values required by the prediction technique to be used. An important final step is continual tracking of the forecast versus actual workload changes. An understanding of any discrepancies shown by such tracking should lead to more accurate forecasts in the next planning cycle.

Sources of the network workload forecast are the user liaison, the network provider, and trend information from the current network.

3.2.1 USER REPRESENTATIVE

In some MIS organizations User Representatives are the liaison between the users and the MIS organization.

The User Representative is in contact with the users of the MIS systems. In such a position the User Representative have access to the users business plans. The User Representative will know the business plans of the users and will be able to help with the forecasting process.

3.2.2 NETWORK PROVIDER

The network provider with the knowledge of the network and the workload will provide input to the forecasting process.

3.2.3 TREND INFORMATION FROM CURRENT NETWORK

The current network will be analyzed using statistical methods to provide growth trends for input to the forecasting process.

3.3 EXPECTED RESULTS

The expected results for this phase will be a better forecast of the network needs. The forecast will be compared to previous actual data to improve the forecasting process.

The expected deliverable is a monthly report that, with the workload characterization report, details:

1. Actual growth vs. Predicted growth by:
   a. Company
   b. Line
2. Predicted information by:
   a. Network
   b. Company
   c. Line
   d. Node
   e. FEP Hardware
3. Trend information by:
   a. Network
   b. Company
   c. Line
   d. Node

Figure 4. Phase 2 Deliverable

3.4 TIMETABLE

The following figure shows the time table for implementing the workload forecasting phase of network capacity planning.

Management Approval	Start
Initial Workload Forecast	+ 2 Months
Workload Forecast with Trending	+ 4 Months

Figure 5. Phase 2 Time Table

4.0 PHASE 3 -- SERVICE-LEVEL OBJECTIVES

Service level objectives are established with the users. Service levels are threshold values that define acceptable performance to the users. This is a crucial activity since the notion of a system being out capacity is undefined until such objectives are set.

4.1 OVERVIEW

For what service-level measures should objectives be set? Typically they are set for timeliness measures such as response time, system availability, network availability, line availability, and node availability. repeatability of charges.

4.2 SERVICE-LEVEL OBJECTIVES

Service levels are threshold values that are measurable and agreed to by the users and MIS. The capacity of the network is undefined until such objectives are set. Some of the issues to consider when negotiating objectives include:

1. What measures should be used,
   a. Network availability,
   b. Terminal Response time,
   c. Line availability,
   d. System Availability,
   e. Maximum Permitted System Outage,
   f. Repeatability of Charges;
2. Should objectives be met on the,
   a. Average,
   b. 90th percentile,
   c. 95th percentile;
3. Different objectives for different times of day;
4. Dollar differences for different service-levels;
5. Management commitment to honor the service-levels.

4.2.1 ESTABLISH SERVICE-LEVEL OBJECTIVES

Service level objectives are established with the users. Service levels are thresholds values that define acceptable performance to the users. This critical activity can only be set by the customers and the MIS organization.

4.2.2 SERVICE-LEVEL MEASUREMENTS

Objectives can be measured in absolutes, averages, or percentages. Absolute measurements do not allow for varying system workloads. Averages measurements do not lead to consideration of peak loading times. Percentages led to considerations of peak loading times. The service-level measurements should be ones that can be measured and are related to the cost of the service.

4.3 EXPECTED RESULT

The expected result will provide to MIS users and the MIS organization a way of measuring how the network is performing and delivering service.

The deliverable is a monthly report on how the network has met the service-level objectives.

4.4 TIMETABLE

The following figure shows the time table for implementing the service-level objectives phase of network capacity planning.

Management Approval Service-Level Objectives	Start + 3 Months

Figure 6. Phase 3 Time Table

5.0 PHASE 4 -- MODELING AND SERVICE-LEVEL PREDICTION

Once service-levels have been established and workload forecasting has been done, then predictive techniques can be applied to determine the equipment needed.

5.1 OVERVIEW

There are two basic approaches to service-level prediction: benchmarking and modeling. The choice of an approach is influenced by many factors including experience of planning staff, time, cost of prediction, and the amount of data available. Usually, it can be said that modeling is less expensive and less accurate than benchmarking.

5.2 TOOLS AND METHODS

The available tools and methods are: rules of thumb, analytic modeling tools, discrete modeling tools, statistical methods, network simulation, and benchmarking.

The use of the models will help in predicting the service-levels that will be delivered.

Modeling tools and techniques ordered by cost are; rules of thumb, statistical methods, analytic models, and simulation models.

These modeling techniques provide a fast low-cost way of predicting service-levels. The capacity planning group may use any or all the above modeling techniques for predictions.

5.3 EXPECTED RESULT

The expected result will be an equipment forecast based on service-levels and workload forecast. The deliverable will be a quarterly report.

5.4 TIMETABLE

The following figure shows the time table for implementing the modeling and service-level prediction phase of network capacity planning.

Management Approval Integrated Capacity Planning	Start + 6 Months

Figure 7. Phase 4 Time Table

6.0 PHASE 5 -- MANAGEMENT REPORTING

The ultimate success of a planning study depends heavily on the reports which clearly state an justify their recommendations.

6.1 EXPECTED RESULT

The expected result of the management reporting part of the project is better informed management. Management that has better information will make better decisions for the company. The deliverables, of the management reporting project are:

1. Monthly Workload Characterization Report
2. Monthly Workload Forecast Report
3. Monthly Service-Level Report
4. Quarterly Equipment Forecast

6.2 TIMETABLE

The following figure shows the time table for implementing the management report phase of network capacity planning.

Management Approval Management Reporting	Start + 2 Months

Figure 8. Phase 4 -- Managemet Reporting Time Table

7.0 SAMPLE WORKLOAD CHARACTERIZATION REPORTS

COMPANY	CURRENT LINES KBS	CHAR. IN	CHAR. OUT	TRANX.	AVERAGE RESPONSE
User ABCD	6 - 9.6				
User EFGH	5 - 19.2 4 - 9.6 1 - 4.8 Dial 1 - 4.8 Dial				
User IJKL	1 - 19.2				
User MNOP	2 - 9.6 2 - 16.8				
User QRST	1 - 19.2				
User WZYZ	2 - 9.6				
MIS	9 - 9.6 2 - 4.8 Dial 1 - 4.8 Dial 2 - ASC Dial				
User 1234	15 - 9.6				
User 5678	1 - 9.6				
User 90AB	1 - 19.2 4 - 9.6				
User 10EF	1 - 9.6				
User 20GH	1 - 9.6				

Figure 9. Network Workload

Form: 50P001

8.0 SAMPLE WORKLOAD FORECASTING REPORTS

COMPANY	CURRENT LINES KBS	FUTURE LINES KBS	DATE	COMM UNITS	ASSUMPTIONS
User ABCD	6 - 9.6	3 - 9.6	1Q87	3	in progress
User EFGH	5 - 19.2 4 - 9.6 2 - 4.8 Dial	2 - 19.2	3Q87	4	in progress
User IJKL	1 - 19.2	2 - 19.2	ASAP	4	in progress
User MNOP	2 - 9.6 2 - 16.8	2 - 56	3Q87	24	
User QRST	1 - 19.2	No Network Growth			
User WZYZ	2 - 9.6	1 - 9.6	end 87	1	Additional Need
MIS	9 - 9.6 3 - 4.8 Dial 2 - ASC Dial	No Network Growth			
User 1234	15 - 9.6	3 - 9.6 30 - 9.6	3Q87	3 30	Forecast Forecast
User 5678	1 - 9.6	1 - 9.6	end 87	1	Growth req.
User 90AB	1 - 19.2 4 - 9.6	No IBM Network Growth			AE Forecast
User 10EF	1 - 9.6	1 - 9.6	end 87	1	AE Forecast
User 20GH	1 - 9.6				
User 301J		1 - 9.6	mid 87	1	AE Forecast

Figure 10. Network Configuration and Needs by User Group

9.0 SAMPLE SERVICE-LEVEL OBJECTIVES

COMPANY	SERVICE LEVELS			
	Response	Avalibility Shift 1	Shift 2	Shift 3
User ABCD				
User EFGH				
User IJKL				
User MNOP				
User QRST				
User WZYZ				
MIS				
User 1234				
User 5678				
User 90AB				
User 10EF				
User 20GH				
User 301J				

Figure 11. Network Service Level Objectives

10.0 SAMPLE INTEGRATED CAPACITY PLANNING REPORTS

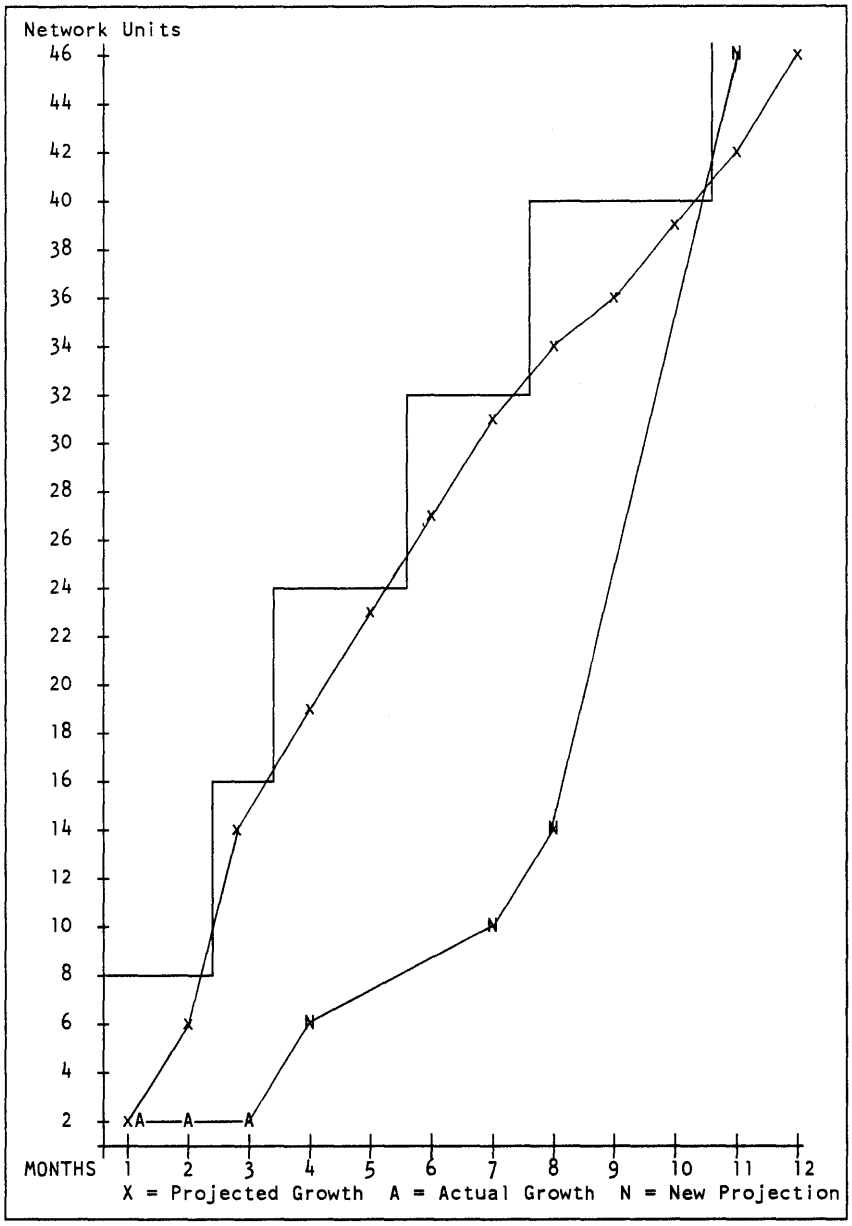


Figure 12. Combined Network Capacity Planning Chart

Form: 50P001

11.0 DOCUMENTATION OF THE NETWORK

Documentation of the network is a requirement for a centralized network control and performance measurement of the system. The documentation should include the following:

1. Circuit layout
2. Network map
3. Hardware/software cross-references
4. Network vendor maintenance records
5. software listings by network task
6. software listings by network component
7. All user site telephone numbers
8. All user site individual contacts
9. Interface component maintenance log
10. Circuit control telephone contact, index and log
11. Maintenance history by component
12. Inventory by serial number of all network components
13. Inventory by serial number by site
14. Network redundancy locations and switching circuits.
15. Vendor contracts
16. Tariff commitments
17. Operations manuals by component.
18. Trouble tickets
19. Escalation procedures
20. Preventative maintenance guidelines
21. Diagnostics techniques by component
22. Diagnostics techniques by type of trouble
23. Disaster and recovery procedures.

The Theory and Practice of I/O Tuning

Rich Olcott
Schering-Plough Corp
3022 Jackson
Memphis TN 38151
(901) 320-2867

System performance concerns may center on maximizing throughput or on minimizing turn-around time. For most installations, either concern is addressed most effectively by minimizing time spent waiting for IO completion, especially for online storage. This optimization can only be accomplished if the system tuner can locate and control the performance bottlenecks in the IO subsystem. Different hardware lines and operating systems supply varying kinds and amounts of such performance-related information. This presentation surveys some useful tools and techniques for extracting and using that information. The discussion centers on the MS-DOS and Apple Pascal systems on microcomputers, UNIX V representing the minicomputer level, and the MVS operating systems run on large mainframes.

It's easy to give an inaccurate description of the performance analyst's objective: "Make the machine run faster". But in fact, the running speed of any given computer system is fixed by its hardware designers. Given a specific CPU, there are only two ways to simulate greater speed: either reduce the CPU work per transaction, or reduce the CPU's wait time per transaction. At least half the job is really "Make the machine wait less".

The charts in Figure 1 summarize a typical day in a typical commercial installation running the MVS operating system. Work processed by four major subsystems (TSO, IMS, CICS, and batch work) was analyzed to evaluate the time spent in various defined states. For example, TSO response times were broken down to average time waiting for a swap-in, waiting for a page-in, waiting for a disk input or output operation to complete, ready but waiting for CPU, actually using the CPU, and miscellaneous waits for such things as device allocation and enqueued resources. These subsystems have very different characteristic patterns of operation, but for each subsystem, the waits for disk response comprised 50% or more of the response or turnaround time. If disk response can be improved, every subsystem will show a performance enhancement. The importance of the input and output operations is so pervasive that the acronym I/O has come into common parlance to refer to generic inputting and/or outputting as specific operations, the subsystem that carries out that work, or the data so transmitted.

The moral of these charts is that the disk farm is usually the most fruitful source of performance improvements. This conclusion can safely be extended across operating systems and to all but the most compute-bound processing environments (and there are far fewer of these than is generally thought). A well-tuned I/O subsystem is the first prerequisite for getting the most out of the CPU that drives it.

Performance concerns in general cluster under three headings: response time, throughput, and availability. Throughput is the number of tasks completed per unit time. Response time (also referred to as turnaround time) for an activity request is the elapsed time between request submission and request completion. Tuning actions that increase throughput will usually decrease response time, but not always. Consider, for instance, that an L-1011 carries many more passengers than a 727, but both require 2-1/2 hours to fly from New York to Chicago. With regard to the I/O subsystem, performance is best treated as a response time problem whereas subsystem capacities are often measured in throughput terms.

Availability is often skipped over as a performance concern, but it shouldn't be. The steady improvement in hardware reliability has had two effects. First, we are less ready and more surprised when failures occur. Second, even less sophisticated systems include provision for detection and correction of some classes of recording and transmission errors. Such activity occurs "under the covers" and can be a significant source of slowed device response.

Performance management includes several distinct types of work, most easily categorized by their time-scales. Trouble-shooting is "right now" activity, oriented to locating and addressing problems in real-time. Online performance measurement tools are indispensible here. Capacity planning, at the other end of the spectrum, is concerned with projecting requirements generated by future workloads. Typical capacity planning tools include surveys of user requirement projections, and packages for modeling and trending calculations.

The word "tuning" covers the middle time-scale, from days to weeks. There is some time for analysis and planning, but not enough to solve a problem by bringing in new equipment. The tuner's tools and options overlap those used by trouble-shooters and by capacity planners, with some additions. In particular, the tuner is the one most likely to be involved with designing device cabling configurations and setting operating system tuning parameters.

An effective system tuning activity has a cyclic structure (Figure 2). We must know our desired performance status and our current status. If they don't match, we make a change (only one!) to the system and loop back to status measurement and comparison. Characterization of performance status depends upon the variables that can be measured under the operating system in question. Desirable items include: I/O wait times and request rates for each workload and device, phase or state times for each device, and error rates for each device and path element. (The system-dependant terms "phase", "state", and "path element" will be defined by example below).

There are two basic strategies for selecting the system change to be implemented. The first depends upon "happy values" or rules of thumb (ROT's). These are guideline target values, derived from experience or calculation, that usually lead to acceptible performance. "Device busy should

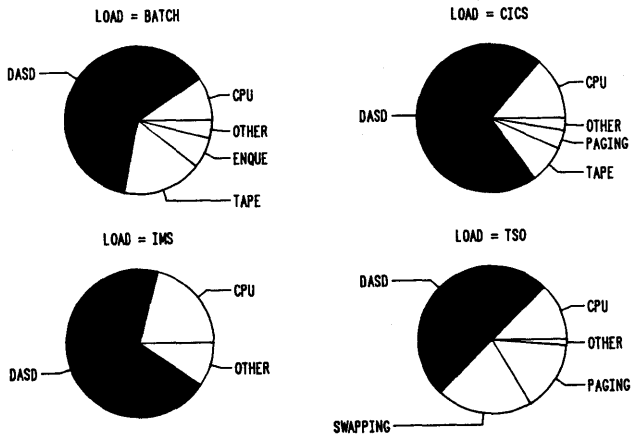


Figure 1. DASD Wait and Response/Turnaround Time

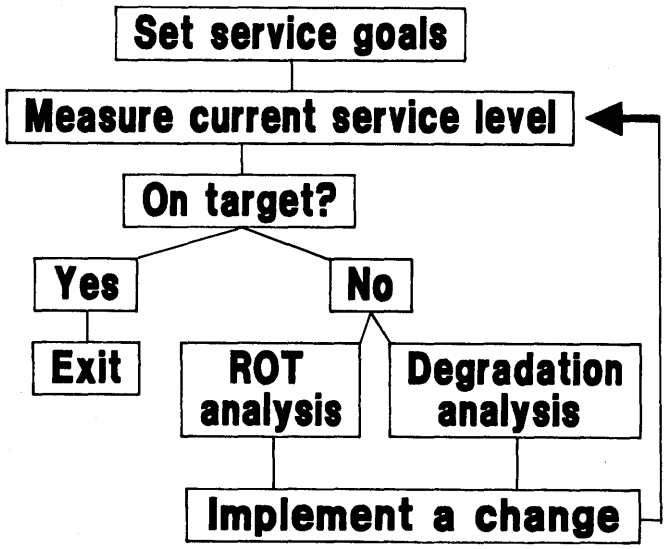


Figure 2. The Structure of the Tuning Process

be less than 30%" is a classic ROT. ROT's are rough averages to be applied to average systems; they do not in general allow for special performace requirements for critical tasks, not do they locate cost savings that could be realized by lowering delivered performance for non-critical work.

The second strategy is more powerful, at the cost of more work in the analysis. Under names such as "degradation analysis", "bottleneck analysis", "time tracking", and "profile analysis", it has been recommended by numerous authors. The heart of the strategy is a structured examination of the life-cycle of a typical or critical transaction. The objective of the examination is to find the system adjustment that will yield the largest reduction of the total response time. The life-cycle is decomposed into wait times and busy times. The times are defined in such a way that each corresponds to a specific tuning action. The longest wait times or busy times are treated first. By concentrating on the most significant components of the life-cycle, one obtains the largest performance improvements early in the process.

The available system adjustments also depend on the system in question. Repair of faulty equipment is always an option. In addition, one can implement operating system tuning options (including buffer allocations), reschedule the work contending for a particular device, rearrange the stored data to minimize contention, and redesign the application.

These general principles apply to computing systems at all levels. To demonstrate this fact and to gain practice at the type of thinking required of a system tuner, we will consider several hardware architectures and operating systems: UCSD Pascal implemented on an Apple II mocrocomputer, MS-DOS on an PClone microcomputer, UNIX V on a minicomputer system, and the IBM MVS/370 and MVS/XA software on a 370-class mainframe [1].

Microcomputer level

The Apple II is a family of microcomputers based on the 6502 chip [2]. USCD Pascal [3] is a single-tasking operating system based on the Pascal language designed by Wirth [4]. The UCSD system includes some language extensions (string operations and a set of I/O procedures), a process control supervisor, and a set of utilities. The Apple implementation of the UCSD system (referred to hereafter as Apple Pascal) contains some additional routines that exploit the Apple II hardware graphics capabilities.

MS-DOS [5] is a single-tasking operating system written by the Microsoft Corporation to run on any of the 8086-based microprocessors ("PClones") that is more or less compatible with the IBM PC family of systems. The Apple and PClone families differ in many respects, but they are sufficiently alike and sufficiently popular that we will use both together as an introduction to I/O tuning in general.

A typical microcomputer system might comprise the CPU chip and memory on a motherboard, a CRT (television or monitor) connected to the system via a display controller card, a disk controller chip on another plug-in card, and one or two floppy-disk drives attached to the controller by a flat cable connector. The essential parts of the disk drive itself are a motor to rotate the disk, a read/write head to interact with the magnetic coating on the disk, and an actuator to move the head radially across the disk.

The data is laid out on the disk in a series of concentric circles, called "tracks". (Numeric data in the next few paragraphs is specific to the Apple system. PClones incorporate drives with somewhat different quantitative characteristics but the same operating principles.) Each track corresponds to a specific radial setting of the head. On Apple II floppy-disk drives compatible with the Pascal system, each of the 35 tracks contains 16 256-byte sectors. Under the Apple Pascal operating system, the unit of file allocation is a 512-byte block comprising a pair of sectors. The internal structure of a file is not defined to the operating system. Although random-access operations are possible, most files are written and read as a sequential byte stream.

The life-cycle of a typical I/O operation starts with a procedure call from the application program. The operating system supplies procedures that read and write a structured record, a delimited line of text, a specified number of bytes, or a full block of data. In all cases, data will be transferred between the disk drive and a block-sized area of main storage known as a "buffer". Many of the input procedure calls can be satisfied by reference to data already in the buffer. In such cases, no physical I/O takes place and there is no delay of the program execution.

Once the system determines that physical I/O is indeed required, the disk controller is instructed to generate the electrical signals that start the disk drive motor. Roughly 500 milliseconds (msec) later [6], the drive is up to rated speed (300 RPM). If the head is not positioned over the proper track, the controller must move it there. This operation, called a "seek", takes a maximum of 400 msec to traverse all 35 tracks. The minimum seek moves the head from one track to the adjacent one. This seek is thus 1/34 of the maximum, or about 12 msec. In fact, delays inherent in head acceleration and deceleration will make the true value a bit more than this. If there is no correlation between one head position and the next, each seek is an "average seek", which can be shown to be one-third of the maximum or about 135 msec on this equipment.

Now the head is over the desired track, but not necessarily over the desired block. The diskette will have to rotate by some angle before that occurs. At 300 RPM, one full rotation takes 200 msec. On the average, one-half a rotation will be required to bring the disk to the proper position. This angular positioning delay, termed "latency", will be 100 msec. With 16 sectors per track, our two-sector block occupies one-eighth of the track; data transfer thus will use up one-eighth of a full rotation, or about 25 msec.

We arrive at the timing data summarized in Table 1. Starting from an arbitrary track on a stopped disk, each request to a random block will require 760 msec.

Suppose our task is to tune an Apple Pascal application to reduce delay to a minimum. Following our tuning cycle strategy, we first ask, "What is the current status?" With

Table 1. I/O Time Constants for a 16-sector Apple II Diskette Drive

Component	Time (msec)
Motor start-up	500
Seek times	
Minimum	15
Average	135
Maximum	400
Latency	100
Data transfer (512-byte block)	25

this hardware and operating system, we have no internally-measured data on application speed or I/O response time. The alternatives are stopwatch measurements, or hardware timing data plus some assumptions. One assumption is that I/O, which runs at mechanical speeds, will always be slower than the CPU work running at electronic speeds. We will proceed on that assumption, after using the stopwatch to measure the initial transaction times.

The largest figure in Table 1 is the motor start-up time. If the drive is given a chance to stop during the compute phase of the work, then the next I/O will be lengthened by half a second. It might be worthwhile to change the application program to do additional I/O, just to keep that motor running. Some system owners have in fact altered their hardware to the same effect. If either change is implemented, we return to our stopwatch study to make sure that there has been an improvement.

Our next recommendation depends upon which is the next largest component of the I/O cycle. That could be either latency or seek time. Both can be affected by manipulating the access pattern across the diskette. Rotation delay would be zero if I/O were timed so that a request always arrived just as the required record came under the head. Several systems (including Apple II and the PClone "standard") have attempted to achieve this ideal for sequential access. The most common technique is to stagger block positions about the track so that block n+1 is two or three blocks past block n. This algorithm succeeds or fails depending on the relative speeds of computation and rotation. If computation runs too long between I/O's, the staggered locations could in fact increase the rotational latency delay.

The key to minimizing seek time is to locate the next block on a track that is close to the current head position. Sequential access minimizes seek time, because all seeks during sequential access are either zero or to an adjacent track. If a file is not to be accessed sequentially, the best tactic is to keep the most popular blocks in the center range of tracks. If multiple files are used simultaneously, it is advisable to dedicate one drive to each file so that a head never has to seek from one to another.

The smallest component of the I/O life cycle is the actual data transfer time. Given a constant blocksize, rotation rate, and data quantity to be moved, the only way to reduce this delay would be to eliminate the I/O altogether. I/O reduction finesses all I/O delays completely. The challenge is to find ways to eliminate I/O activity when, as is often the case, the software requesting the I/O is a purchased program that cannot be altered.

One general strategy is to read often-referenced data once and maintain it in a spare buffer instead of re-reading it. This technique reduces I/O at the expense of main storage and CPU cycles. Both the Apple Pascal system and MS-DOS provide buffer re-read and buffer write defer, in the expectation that the data already in the buffer may still be interesting or may receive yet another update. How many buffers are available for this purpose? Apple Pascal's buffer count (not published) was established when the operating system code was compiled at the factory. MS-DOS is more flexible: the user can optimize the resource trade-off for his/her application by setting the value of a parameter (the BUFFER= command in CONFIG.SYS). The default is 2 or 3, and most users find 10-15 a good compromise between I/O speed and memory encroachment.

The MS-DOS directory system is a good example of an "application" whose I/O can be optimized even though the software itself is beyond the control of the tuner. The directory serves as the table of contents to the files on a floppy or hard disk. No file can be accessed without going through the directory, so slow directory I/O impedes all other work. How should a performance-conscious user treat the directory to minimize wait time?

A single MS-DOS directory on a 360K floppy can contain information on up to 112 files. (The limits vary for other media.) A single disk can contain many times that number of files. MS-DOS addresses the disparity by providing for a hierarchy of directories and sub-directories (Figure 3). Each directory is composed of one or more clusters of 512-byte data blocks. The number of blocks in a cluster varies with the recording medium and operating system level: 1 block per cluster on a floppy, through 16 blocks for a 10 megabyte hard disk and DOS 2.x.

The directory clusters are allocated just like other clusters on an MS-DOS disk, on a first-come-first-served basis starting with the lowest-numbered unused block. For example, the first block of the root directory will automatically be located at the first data block position (following system-specific elements like bootstrap records and a space accounting table). The next block within the first cluster will be immediately adjacent to the first. The next cluster within the directory will not be allocated until after some other files have been built on that disk, so the second directory cluster will be positioned well away from the first. Passing from the first to the second cluster within the root directory generally requires a long seek. Lesson number one: if the cluster size is 2048 (typical but not imperative for a hard disk), you avoid intra-directory seeks by putting no more than (2048 / 32 bytes per directory entry) = 64 files in a directory. (Notice that "file" includes sub-directory pointers as well pointers to commands and data files.)

It should be clear that inter-directory seeks also generate delays. From a performance standpoint, the ideal directory structure is broad, rather than deep. The ideal would be to have a root directory which contains the most often-used commands (to minimize time getting to them) and pointers to sub-directories. No sub-directory would contain a pointer to any lower-level, sub-sub-directory. No directory would span more than one cluster. To bring directory I/O delay to the minimum, all directories would be declared before any other files are built on the disk, because that procedure places all directories together next to the space allocation table.

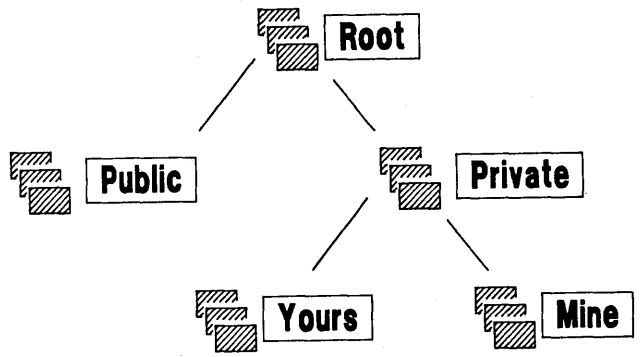


Figure 3. A Sample MS-DOS Directory Structure

UNIX V

The UNIX operating system [7] adds several layers of complexity to our tale. Apple Pascal services a single user at a single terminal, supporting but a single task. UNIX is designed to support multiple users at multiple terminals, with multiple asynchronous tasks running concurrently. Furthermore, the Apple Pascal system was designed to exploit the capabilities of a specific piece of hardware; UNIX has made a virtue of its hardware independance. In fact, the UNIX I/O architecture is so abstract that it would not be worthwhile to describe its interaction with any specific disk device.

A UNIX disk is formatted into 512-byte blocks. A UNIX file is a sequence of blocks linked together by chain pointers. There is no necessary relation between the physical locations of blocks that are logical neighbors within a file. Files written onto a freshly initialized disk may well be placed in contiguous storage. As blocks are used, then given up and reused, the odds increase that successive blocks in any given file will be widely scattered across the recording surface. In general, we can expect that all seeks on a mature UNIX disk will be "average" seeks, taking about 1/3 the time of a maximum seek on the device in question. Furthermore, there is not a reliable way to reduce rotational latency, because (1) there is no regulation of relative block position, and (2) there is no way of predicting compute time between I/O's in the presence of concurrent tasking.

Systems giveth as systems taketh away. The UNIX support of multitasking adds an important source of information to our tuning tool kit. One program can monitor and report on the performance of other programs as well as the system as a whole. UNIX V has several such facilities.

The System Activity Data Collector (command /sadc) maintains several sets of counters in memory. One set of counters records events such as completed I/Os and task switches. These counters are maintained by a portion of /sadc that receives control during system calls. Another set of counters is maintained by a portion of /sadc that records the state of various entities including the processor and I/O devices. This /sadc process wakes up periodically. At each awakening, it scans the system to see if, for instance, a particular device is servicing an I/O. If so, the corresponding counter is incremented. The command also checks to see if other processes are waiting to use the device; the number of waiting tasks (the "queue length") is added to yet another counter assigned to that device.

Performance monitors like /sadc can soak up significant system resources if they attempt to do too much. The UNIX designers wisely chose to limit /sadc to data collection only, reserving data reduction and presentation to other commands. The System Activity Reporter (/sar) and the System Activity Grapher (/sag) are the output side of the /sadc system. Each can generate displays of a wide variety of measured quantities. They differ in mode of operation: /sar produces tabular reports and a binary file; /sag reads that binary file in order to produce its graphs.

The /sadc system measures quantities on a system-wide basis. Two additional commands provide data with a narrower focus. The System Activity Disk Profiler (/sadp command) is a tool for examining the seek pattern on a specific disk. During the measurement interval, each of /sadp's samples yield one observation of the track addressed by that device's read/write head. Over a period of time, these addresses form a picture of the most popular areas of the recording surface. Undesirable patterns, such as widely-separated or very broad peaks, show up readily on the /sadp-generated histograms.

The other major tool is the /timex command, used to investigate the performance characteristics of a specific process. In effect, /timex "wraps around" the selected program, recording for it alone the same set of variables recorded for the entire system by /sadc.

With tools in hand and measurements on the desk in front of us, how do we decide what looks good or bad and what to do next? To begin with, we might use the ROT approach, looking at the system-wide numbers given us by /sar. In general, we'd like to see %busy spread evenly across all disk devices. On the average, %busy for disks should not be greater than 30% (tapes can go much higher). Desirable values for avserv are device dependant, because the service time includes seek, latency, and data transfer times. Numbers for hard disks, for instance, are very much shorter than for floppies. We can get an idea of what to expect by adding together the vendor's specifications for average seek, one-half the rotational period, and (512 divided by the rated data transfer speed). If the observed avserv for a device is consistently more than 1.5 times this sum, it is a good candidate for further investigation.

I/Obuffers are maintained in a region of main memory termed "the cache". A block once read into memory remains in the cache until it is the oldest buffer and some process needs one for a different block. If a block already in memory satisfies an additional data request (read or write), the event is called a "cache hit", representing the avoidance of a physical I/O. The cache is being well-exploited if %rcache is greater than 90% and %wcache is greater than 65%. The latter value is highly application dependant and may not be approached by programs that force physical writes either by direct system calls or by frequently updating large records.

Table 2. Selected UNIX V Measurements

Device activity
- %busy - Device utilization
- avque - Average queue length
- avwait - Average wait time
- avserv - Average service time
- r+w/s - Data transfers / sec
- blks/s - Blocks transfered / sec

Swapping activity
- swpin/s - Swaps in / sec
- swpot/s - Swaps out / sec
- bswin/s - Blocks swapped in / sec
- bswot/s - Blocks swapped out / sec
- pswch/s - Process switches / sec

Directory activity
- iget/s - System path searches / sec
- dirbk/s - Directory READs / sec

System calls
- sread/s - READ calls / sec
- swrit/s - WRITE calls / sec
- rchar/s - Bytes read / sec
- wchar/s - Bytes written / sec

Buffer activity
- bread/s - Physical block READs / sec
- lread/s - Logical block READs / sec
- bwrit/s - Physical block WRITEs / sec
- lwrit/s - Logical block WRITEs / sec
- %rcache - READ hit ratio
- %wcache - WRITE hit ratio

CPU activity
- %usr - % of time in user mode
- %sys - % of time in system mode
- %wio - % of time waiting for I/O
- %idle - % of time idle

System table activity
- text-sz - Text table size
- ov - Number of overflows

On the other hand, hit ratios much higher than these may signal too large a cache size; reducing the number of buffers may speed processing by increasing the amount of main storage available for computation. Such over-commitment of memory will show up in a comparison of swpin/s and swpot/s with pswch/s. Another clue is a high %swpocc value, indicating a large percentage of processes that are swapped out but otherwise ready to run.

Various authors have given ROTs for some of the other /sar variables. For instance, %wio greater than 30% probably indicates excessive I/O delay. However, most of these are critically dependant on the relative speeds of the CPU and the disk equipment.

Degradation analysis begins by asking "Is there pain?" and if so, "Where DOES the time go?" User complaints of slow response time are occasionally valid, but objective measurements leave less room for debate. We could resort to the stopwatch once again, but UNIX gives an alternative. In some installations, tuning effort commences only after the user has presented a log showing the process in question bracketed within a shell procedure between two /date commands. In this way, the start and end of execution are each timestamped and the response time can be calculated to the second.

Assuming that the response time is indeed unacceptably high, the next step is to rerun the process under the control of /timex. Comparison of %usr, %sys, %wio, and %idle quickly shows whether I/O time is a major contributor to the execution time. After all, it is possible that the process is being delayed by contention for the processor or some other resource. The product of %wio and the observed execution time is the maximum amount of time that could be recovered by I/O tuning alone. If %wio promises useful dividends, we turn to %rcache and %wcache to see if I/O waits could be avoided by avoiding I/O. The hit ratios will in general increase if the cache is enlarged, which can be accomplished by adjustment of the NBUF system tunable parameter.

If I/O cannot be avoided, we look next to see where it can most effectively be sped up. The expression (avwait * blks/s / 1000) gives the fraction of the execution time taken up waiting to start an I/O on the reporting device. Similarly, (avserv * blks/s / 1000) is the time fraction during which I/O was in progress on that device. Wait times lengthen if service time is long compared to the arrival rate of new requests. Either service is particularly slow, or the multitasking is allowing significant contention for the device. Thus, if both total wait and total service times are long, it's best to look first at factors affecting service time. If waits are long but service is swift, we must address contention by offloading data from the contended disk or else by rescheduling the contending work.

Long service times are almost always the result of excessive seek activity. The /sadp command will help us locate the busy tracks; other system utilities can then be used to determine which files reside on those tracks, and then to move these busy files to contiguous regions of the surface.

MVS

Because of IBM's dominant position in the mainframe equipment market, its more successful products have become de facto standards for generic device classes. Thus, specifications for recording surface geometry (number, arrangement and size of tracks), timing constants and command protocols for a specific IBM disk drive type actually apply to a class of drives produced by several manufacturers. In what follows, mention of a 3350-type drive, for example, denotes any drive meeting the corresponding geometry and timing specifications. Similarly, the phrase "370-class" denotes equipment (computer systems, disk drives, etc.) that comforms to the IBM 370 architecture specification.

MVS/370 and MVS/XA [8] are the most-used operating systems on 370-class computing systems. Both are primarily throughput oriented (for batch work) but can be optimized for rapid response to interactive transactions. Whereas the Apple system serves a single master and UNIX serves 2 to 20, MVS systems can handle hundreds (under TSO) to thousands (under IMS). Whereas an Apple may be equipped with 1 to 4 disk drives, a mainframe system may contain hundreds of disk drives, each holding up to 9000 times the data of an Apple Pascal floppy disk. Such increased system capacity comes at the price of commensurately increased system complexity.

MVS/XA is descended from MVS/370. There are numerous differences between them, but for our purposes there is only one. In MVS/370, all the work of processing an IO request is done by the CPU. In MVS/XA, much of that work has been offloaded to subsidiary processors located between the CPU and the disk subsystem. There are several implications of this difference. First, of course, more of the power of the CPU is available for application computation. Second, the system gains in flexibility, reliability, and responsiveness, because individual IO requests can be more closely managed (re-routed, clocked, etc.) by the specialized outboard hardware. Third, the CPU is less able to record and report on exactly what is happening outside of its view.

There are two architectures of 370-class disk drives, corresponding roughly to the hard-sector/soft-sector classification of floppy disks. The hard-sector group, defined by the Fixed Block Architecture (FBA), is rarely used in MVS installations and will not be considered here. In the more common architecture the recording unit of data is a block whose length is set by the application. As with the simpler systems, blocks are arranged in concentric tracks on a rotating recording surface. Unlike the simpler systems, a 370-class drive will contain a stack of corotating surfaces, each with its own read/write head. Because there is only a single actuator to move the entire set of heads, an entire stack of tracks will be accessed by a single actuator setting. It is convenient to refer to this track subset as a "cylinder".

In the Apple system, the disk controller chip does the translation of the CPU's disk access commands into electric signals such as "start motor" and "move actuator inward by five steps". The 370-class equivalent to the controller chip is actually a series of complex mechanisms (Figure 4). The disk drive itself contains only the recording surfaces, the motor, the actuator, and the heads. A string of drives is managed by a "head of string" (HOS), which contains electronics for device polling, data transmission, and the lowest level of command translation. One or more strings is cabled to a controller, which does higher level translation and may contain memory (a "cache") of its own. The controller is in turn attached via a "channel" (several different terms may apply, depending on the specific processor in question) to the CPU. The entire series, from channel to device, is called the "path"; each component of the path is a "path element" (PE). During data transfer, each component is dedicated to that particular IO event. Contention for PEs is a major source of IO delay.

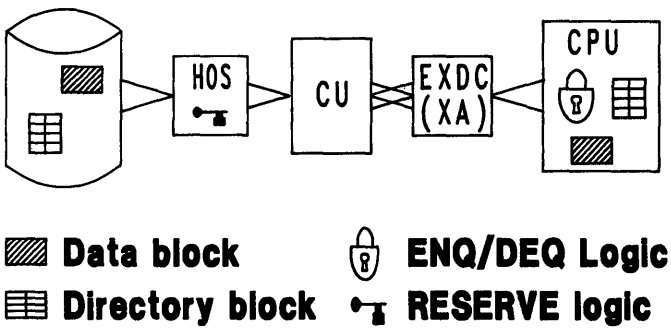


Figure 4. IBM Mainframe DASD Architecture

To exploit the random-access capability of disk hardware, IBM and other vendors have provided a number of "access methods" (AM's). An AM consists of a particular structuring of the data within a block, together with protocols and software to interface application programs with that data. Available modes of access range from simple sequential (forward and backward) through skip sequential and simple keys, to highly sophisticated retrieval strategies based on binary trees or hash techniques. Much of the software is devoted to generating the disk drive commands (the "channel program") that exploit the processing capabilities of the channels and control units. The following discussion will concentrate on the Queued Sequential Access Method (QSAM), which is quite similar to Pascal structured IO. Under QSAM, each block on disk contains one or more records. Records and blocks may be of fixed or variable length, and a record may extend across one or more blocks. It is QSAM's responsibility to keep track of record boundaries.

A typical IO sequence begins when an application issues a GET request to the QSAM software, asking for the next record in a particular file. QSAM owns in-memory buffers dedicated to this file. Because of a previous READ operation, the block containing the next record may already be in a buffer. In that case, the GET request can be satisfied without doing additional IO. If an IO is required, QSAM builds a READ channel program. The program points the channel to the desired device and block, and directs that the block be placed into a specific buffer. The channel program is passed to the IO Supervisor (IOS), which is responsible for overseeing the successful completion of the request.

IOS's first action is to verify that the device in question is free to transmit data. In a multi-programmed, multi-processing system like MVS, many processes may run concurrently so another process may now be engaged in IO with that disk or another one on the same path. Rather than initiate a doomed activity, IOS queues our request until the device and path appear available. At that time, IOS issues a SIO (for MVS/370) or SSCH (for MVS/XA) command to initiate the waiting channel program. Initiation may in fact not succeed, because some element of this path may have been claimed by another IO from this CPU or another that may share the device or path. In MVS/370, our IO request goes back on the IOS queue. In MVS/XA, the request waits in the outboard processor.

Sooner or later, the request is finally able to start. The first command in a QSAM channel program is a SEEK, telling the actuator to move to the appropriate track. The next command is a SET SECTOR, specifying the starting position of the block to be transmitted. The last command is the READ, specifying the number of bytes expected. Notice the device timing constants collected in Table 3.

Rather than tie up PEs during mechanical movements, the IO hardware itself disconnects the path from the drive until the movement operations complete. Now the disk attempts to reconnect to the path in order to receive the next command. (In 3380-type devices, SEEK and SET SECTOR are

Table 3. I/O Time Constants for 370-class Disk Drives

Component	Times (msec)		
Device type	3330	3350	3380
Seek times			
Minimum	10	10	3
Average	30	25	16
Maximum	55	50	30
Latency	8.4	8.4	8.4
Data transfer (4K block)	3.3	3.3	1.3

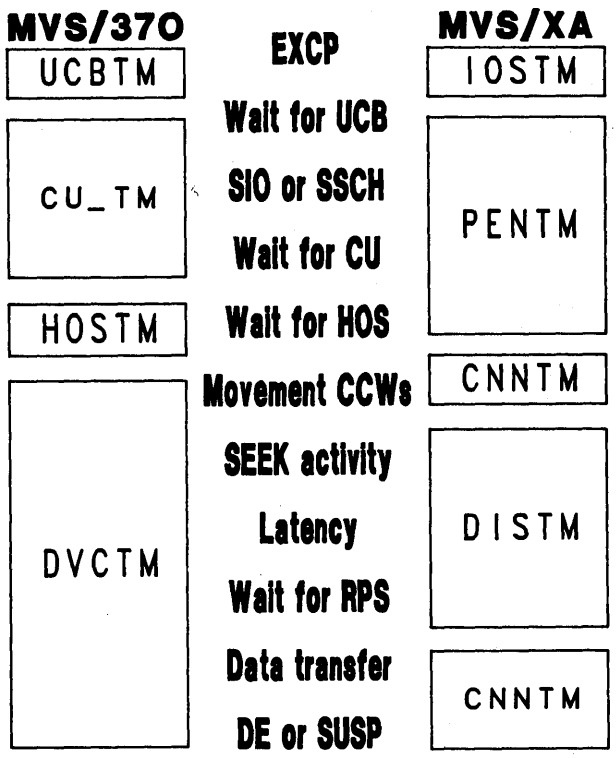


Figure 5. Measuring an MVS I/O Event

transmitted together.) If the path has become busy, the device must wait until the blocking PE is freed. This wait is particularly important at the end of the SET SECTOR operation. At this time, the disk has rotated so that the desired block is about to come under the head. If path reconnection is unsuccessful, then the disk must make a full revolution before reconnection can be attempted again. What had been an 8 msec rotational latency has become a 25 msec delay. The hardware facility that permits the disconnect/reconnect cycle is called Rotational Position Sensing (RPS); the additional rotation because of an unsuccessful reconnect is termed an "RPS miss". If the path is quite busy, a single IO request may suffer multiple RPS misses.

At long last, data moves up the path. However, suppose lightning struck nearby during data transfer. Spurious signals were added to the transmission. 370-class systems have two layers of error detection and error correction capability, residing in the path and in IOS in the CPU. If the error can be corrected in the path, the only evidence is a lengthened apparent transmission time and a "temporary error" entry in the system log. If IOS detects the problem, it attempts to recover by generating a number of retries of the entire channel program. Again, if recovery is successful the only trace of the problem is lengthened times and a log entry for a "temporary error". If recovery is unsuccessful, the application will be abended and the system log will show that a "permanent error" has occurred.

All this complex activity would be unmanageable without appropriate measurement tools. The multi-tasking nature of MVS allows monitors to observe and report on many aspects of system behavior. Several vendors have products to aid the MVS system tuner; we will discuss them in a generic fashion after we have considered the tools that are included in the operating system itself.

The most general of these are the System Management Facility (SMF) and the Resource Measurement Facility (RMF). SMF monitors and records execution times and resource utilizations on a task-by-task basis. One or more records are produced for each batch job, TSO user, and started task in

the system. The records are placed in a special file by a dedicated SMF AM. IBM and other vendors have taken to using the SMF file and AM as a central repository for audit and monitor data for a variety of purposes.

Base RMF consists of a pair of monitors and several reporting mechanisms. The monitors watch over all the system resources: CPU and peripheral devices and several categories of summarized workload resource utilizations. The exact variables differ between MVS/370 and MVS/XA systems. In MVS/370, the CPU is aware of the progress of each IO operation. At any moment, a given PE is known to be in a certain state (busy or not busy, with a certain number of waiting requests) and a given IO request is known to be in a certain phase (waiting for device, waiting for control unit, busy on device, etc.). RMF/370 is a sampling monitor like /sadc, maintaining a set of counters for each device. The system tuner specifies an RMF sampling interval and a recording interval. The monitor routines wake up once each sampling interval to record the state of each device and the phase of any IO request(s) to the device. At the end of each recording interval, RMF builds an SMF-type record containing the current values of all the state and phase counters, which are then reset for the next recording interval.

In MVS/XA, on the other hand, the CPU knows only that an IO request has been passed to the channel. The states of the PEs are not available to it, nor is the current phase of the IO request. On the other hand, the XA hardware is able to measure and report the amount of time spent in each phase at the completion of each request. As a result, RMF/XA device performance data consists of clocks. Figure 5 shows the events of an MVS IO cycle, together with a comparison of the RMF/370 and RMF/XA measured variables.

The first delay is the same in either system: IOS will not attempt to start an IO request against a device when it knows that the device is already busy. Neither system's RMF reports show this delay directly. Both, however, report the IO request rate and the average queue length. The queueing delay can be calculated from Little's Law as:

IOS wait time (sec) =
(Ave queue length) / (IO requests / sec)

RMF/370 reports state percentages rather than wait times. The times can be calculated from these percentages by formulae such as those given in Table 4. RMF/XA reports the times as measured by the outboard hardware. In general, "pending time" is the total path wait time, equivalent to the sum of of the waits for control unit, HOS, and time lost because the device is held by a sharing processor. "Disconnect time" is the total time spent waiting for mechanical motion. "Connect time" is protocol time, data transfer time, and error recovery time. Hardware at and above the IBM 3090 level can measure and report separately on control unit and HOS wait times.

The two RMF monitors differ in time range covered. RMF Monitor I runs whenever RMF itself is running. Monitor I continuously records workload activity and processor loads as well as the IO data we are discussing here. Monitor II is a snapshotting facility, periodically recording instantaneous or short-time utilization levels and counter values. There is a good deal of overlap with the types of data covered by Monitor I, but the time-range difference can itself be useful.

As stated above, RMF has several reporting mechanisms. An online facility queries the Monitor II data collector and can generate any of a set of pre-formatted displays. These displays show current conditions only, and are best used for

Table 4. RMF DASD Performance Fields, RMF/370
Field Names refer to SMF Type 74 (Device Activity) Record

```
DURATION = 60 * FLOOR (SMF74INT/100000) +  MOD (SMF74INT,100000)/1000

SIO Rate:
PSSIO = SMF74SIO / DURATION                    per second

Average queue length:
AVQUE = SMF74QUE / SMF74SAM

Total time (msec) in state xxx during sampling interval:
xxxTM = 1000 * SMF74xxx * (DURATION/SMF74SAM)
         xxx = QUE      Request on IOQ for any reason
         xxx = BUS      Request on IOQ because dvc busy this system
         xxx = ACT      Device busy this system

         xxx = RSV      This system held a RESERVE
         xxx = RDL      Some request held for HOS or external RESERVE
         xxx = DVB      Some request held with dvc busy this system
         xxx = PTH      Some request held with dvc not busy this system
         xxx = CUB      CU busy (any system) on another request

Average queued wait (msec) (Little's Law):
AVQTM = 1000 * SMF74QUE * (DURATION/SMF74SAM) / SMF74SIO

Average service time (msec):
AVSTM = 1000 * SMF74ACT * (DURATION/SMF74SAM) / SMF74SIO

Average response time:
AVRTM = AVQTM + AVSTM

Average wait (msec) at UCB:
UCBTM = 1000 * SMF74BUS * (DURATION/SMF74SAM) / SMF74SIO

Average wait (msec) at CU:
CU_TM = (AVQTM - UCBTM) * SMF74CUD / (SMF74CUD + SMF74RDR)

Average wait (msec) at HOS or RESERVE:
HOSTM = (AVQTM - UCBTM) * SMF74RDR / (SMF74CUD + SMF74RDR)
```

trouble-shooting. In particular, the displays highlight overloaded paths and devices, and provide indications of contentions for critical resources like page datasets and catalogs.

Another RMF reporting system places up to ten different types of records into the SMF file. These records can be used by the RMF post-processing report generator to produce a variety of listings suitable for trend analysis and problem determination. The record formats are described in the vendor's literature, so they can also be read by user programs.

There are several additional sources of performance data in the base operating systems. One of the most important is probably also the most neglected. This is the Error Reporting Program (EREP), a part of IBM's hardware support. This program reads the system log into which the hardware places notations of device malfunctions. Since temporary errors lead to long response times, and permanent errors lead to outages, the EREP reports should be scanned regularly by the tuner as well as by the computer repair technician.

The most useful EREP report is the Trends report. This listing contains a detail line for each peripheral in the system. Shown along each line are the number of permanent and temporary errors suffered by that device for each day in a 30-day period. This "map" layout highlights those devices that should be repaired (permanent errors) or from which at least performance-critical files should be offloaded (temporary errors). There is some controvery about the treatment of temporary errors. Several studies have shown little correlation between temporary and permanent errors. The conclusion is that "weak bits" probably don't "grow" into unusable ones. If that is the case, then one must consider the trade-off between slower disk response versus complete loss of use of the device during repair and adjustment.

MVS actually maintains two system logs. The first, LOGREC, is the one used by EREP and is used mainly for error and malfunction management. The second log, SYSLOG, is essentially a copy of system console traffic. It contains all system operator commands and the system's response to each. It also contains notations of the start and completion of all batch jobs and started tasks. Other information that can be placed in SYSLOG include such items as distress alerts from the telecommunications or other subsystems, status messages from major applications, and indications from task abends. SYSLOG is an important source of answers for the "What else was going on when things went sour?" question.

There are four general classes of performance data sources that do not come along with the MVS operating system: extended monitors, extended post-processors, SEEK analysis systems, and modeling packages. Each addresses a different class of problems. The extended monitors are most useful on the trouble-shooting time-scale. Probably the most sophisticated is the Candle Corporation family of products based on their Omegamon offering. This is basically a system operator's tool. Rather than simply enable an operator to inquire about the current utilization level of a specific resource, Omegamon itself continually monitors all resources. If any utilization goes over a pre-determined threshold, the software displays an alert message on a dedicated terminal. This exception reporting philosophy allows the operator to pay attention to what's wrong rather than having to cope with a flood of data about what's right.

Monitor extensions either add a wider range of managed variables, or lengthen the reporting time-range, or both. One set of new variables is especially important here. The Candle Dexan product and IBM's RMF Monitor III (available only under MVS/XA) both concentrate on workload delay rather than on resource utilization. Each named task in the system is sampled periodically. The task is classified as working, idle, or delayed waiting for some event or resource. Sampled status counts are converted back to percentages over the most recent group of samples. The analyst can then call on the monitor's associated reporting tool to ask for displays of which tasks suffered the most delays, what they were waiting for, and which other tasks were contending for the delaying resource. These products cannot be ignored by an installation seriously interested in effective and efficient solution of day-to-day performance problems.

One of the major shortcomings of the base RMF online reporter is its lack of memory. Many trouble-shooting problems would be most easily solved by comparing the current situation with the one existing 20 minutes ago, say, or maybe yesterday. The Candle Epilog product, Boole and Babbage's CMF system, and the IBM RMF Monitor III all addresss this problem by providing some sort of memory capability, usually extending back only some hours or to the most recent IPL, whichever is less.

These tools are very useful on the trouble-shooting time-scale, but provide little help to the tuner and capacity planner who are more concerned with trends acting over periods of weeks to years. These needs have been addressed by several systems that read records in the SMF file and produce a database that can be maintained to represent the time periods of interest. Examples of such systems include Morino's MICS product family, SAS Institute's MXG system, and a portion of the FastDASD system now being marketed by UCCEL. The Boole and Babbage CMF system also includes a performance database fed from that company's proprietary system monitor. At minimum, these databases allow us to track system performance numbers for exception reporting and management summaries. In fact, their full utility is limited only by the imagination of the people who work with them. We will see some illustrations below.

The third group of software tuning products consists of specialized disk management tools addressing SEEK delay. Another portion of UCCEL's FastDASD product is a SEEK monitor and analysis system, as are Boole and Babbage's DSO product and a free-ware system available from the SHARE user group. Each is fundamentally similar to the UNIX /sadp facility: a monitor routine periodically samples the head position for all or a selected subset of the disk devices attached to the CPU running the software. The motion of the head is assumed to be fairly represented by the samples. These products supply one thing that /sadp doesn't: they also read the subject disk's volume table of contents (VTOC) and so are aware of the location of every file on the disk. Thus, they are able to use the head movement data to make recommendations for file relocation.

The fourth major category of follow-on tools consists of the modeling packages. The oldest generally available system is the Best/1 system from BGS, followed by Boole and Babbages's CMF and Amdahl's MAP. The modeling tools come into the tuning cycle after we've decided what's wrong but before we've decided what to do about it. A model provides no new insight into the currently running system. Rather it permits us to play "what if" games, seeing what effect a given change will produce. A model is used to compare the probable results of alternative solutions to the problem: move this file to that other volume, shorten this string, add cache to that controller. The models help us to rule out solutions that won't solve the problem, and also to put numbers to the inevitable cost-benefit trade-offs to be considered.

Now that we know how the hardware and software operate, let's enter the tuning cycle. We use the measurement tools and obtain a snapshot of the system. Now we need to decide if something's wrong, and if so, what to fix. Let's look first at the ROT approach. Are there any quick indicators of the overall health of the IO subsystem? Yes, and a collection of them appears in Table 5. However, we must keep in mind that a healthy IO subsystem does not necessarily imply that critical work is not being delayed within it. It is risky to rely on ROTs out of the context of service level goals. If critical work is not being delivered in a timely manner, than

Table 5. Rules of Thumb for MVS I/O Tuning

Measurement	Value	Notes
Both systems		
Device busy - batch	40%	
- response	35%	
PE busy	35%	45% for 3380 under XA
Device avg service time	30 msec	3350
	25 msec	3380
Device activity rate	18/sec	3350
	22/sec	3380
Avg queue length	0.10	
% Device RESERVEd	5%	Except JES2 Primary Chkpt
MVS/370		
% Chan Busy & CPU Wait	10%	
Channel avg service time	10 msec	
% Req Deferred - Path	25%	
(%delay) / (dvc busy)	30%	%delay = % reserve delay + % CU delay + 100 * (Avg q len)
MVS/XA		
Average IOS time	5 msec	
Average pending time	2 msec	
Average disconnect time	20 msec	3350
	15 msec	3380

the system needs tuning even if the IO averages are acceptible.

Probably the most famous ROT is the 35% rule for disk devices and PEs. Most tuners try to keep the activity spread out evenly across all drives, and the %busy for each less than 35%. Similarly, the %busy for all controllers and HOSs should be lower than about 35%. However, advancing technology has relaxed some of the constraints that set those limits. For XA systems, the %busy for PEs servicing 3380-type drives (not for the drives themselves!) can go up to 45% before performance starts to suffer significantly. Notice, by the way, that PE %busy is not the sum of %busy for each end device on the path. In XA terms, the path is only busy during connect time, so XA PE %busy can be calculated from the ratio of path total connect time to measurement duration. In MVS/370, on the other hand, explicit PE %busy cannot be calculated. For ROT purposes, we use the Physical Channel %busy from the RMF Physical Channel Activity Report.

There are two other commonly-used ROT's, one measuring load and one measuring stress. The load on a device is roughly proportional to its IO rate. Stress develops when a system component cannot service requests as rapidly as they arrive. That device's average queue length starts to increase, and a length greater than 0.10 is probably a sign of trouble.

The matter of device sharing is a paper unto itself [9]. The 370 architecture allows two or more processing systems to share access to a common disk controller and the device strings attached to it. The arrangement opens numerous exposures in data integrity, security, and performance. The RESERVE is a hardware feature that can be used to preserve integrity. Any processor can issue the RESERVE command as part of a channel program directed to a particular device. The path HOS then blocks IO requests to that device from any processor other than the one holding the RESERVE. The other processors get back only an indication that the device is busy at HOS. This condition lasts until the holding processor issues an RELEASE command. Data integrity is thus bought at the expense of data availability and perceived device performance.

RESERVEs held while the processor does IO to the device are harmless: other processors would be blocked from the device anyway. The MVS code that manages the device storage space, for instance, uses RESERVE during brief periods of intense IO with no ill effects. However, other utilities and applications may not be such good citizens. The 5% ROT serves to alert us to potential abusers of the feature. Note that heavy RESERVE activity is an essential part of normal operation for a JES2 primary checkpoint volume.

ROTs peculiar to MVS/370 IO systems refer to the numbers that appear on or can be calculated from the RMF/370 reports. The most critical to total system throughput is probably the fraction of the time that the CPU is waiting while a channel is completing an IO request. The most telling indicator of structural problems in the IO subsystem is the fraction of all deferred requests that were delayed because of path contention.

ROTs are of much less use in the XA environment, because performance statistics are reported in terms of actual delay times. This reporting convention leads more naturally to a degradation analysis approach than to one depending on ROTs. However, the table does carry a few threshold criteria more suitable for rapid screening than for diagnosis.

As graphics hardware becomes less expensive and more widely available, graphics methods will become more important in system tuning. Figure 6 is an example of the sort of extended ROT technique that can be expected to proliferate in the near future. In this contour plot, time runs along the horizontal axis. The vertical axis has one location for each disk device attached to the reporting system. The z-coordinate, aimed up out of the page, may be any performance statistic. In this example we see the mean path wait time per SSCH. The diagram is thus a depiction of activity or delay across the entire disk farm throughout a week. This sort of display puts an enormous amount of detailed

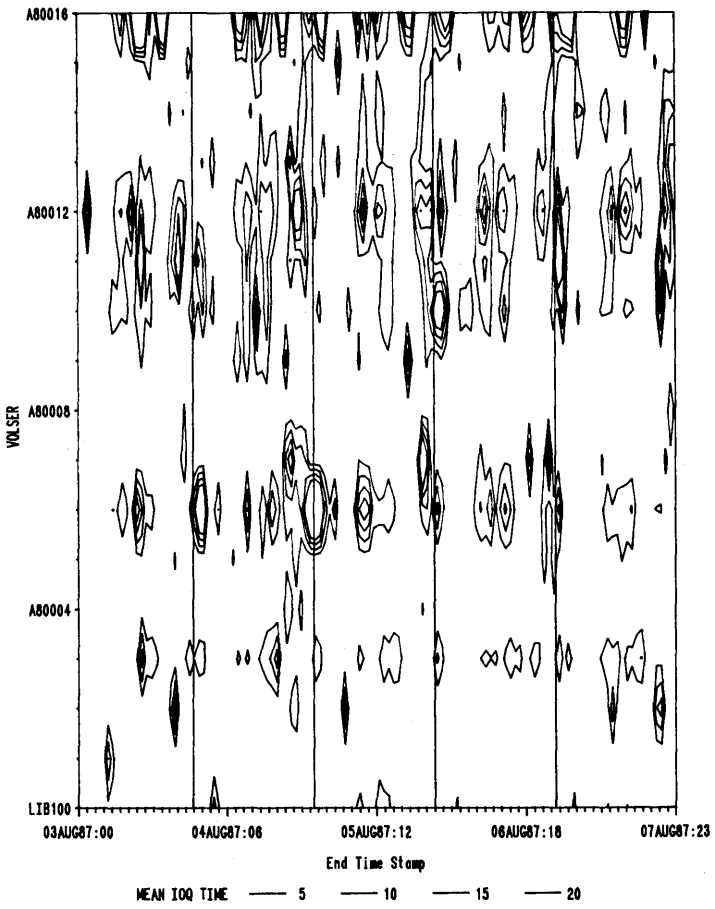


Figure 6. A Sample DASD Delay Map

information into a form especially well-suited to the pattern detection capabilities of the human eye. "Hot spots" show up clearly, as do any time-correlated behaviors.

As we said above, degradation analysis is more readily carried out under MVS/XA than MVS/370. MVS/370 statistics can be used to derive wait times at various queueing points, but most of the arithmetic must be dine by the user. A complete set of formulae has been given elsewhere [9]; a convenient subset is laid out in Table 4. A thorough analysis requires some programming for data reduction and transformation. For convenience, we will state the rest of this discussion in terms of the MVS/XA variables.

As always, the first step is to assay the current state of the system and any critical workloads. The SMF Processing Data record group (types 4, 5, 6, and 26 or else type 30) supply the needed turn-around time data for batch work. Averaged TSO response times can be obtained from the RMF Workload Activity reports or Type 72 records; more sophisticated analyses will require a transaction classification tool such as the Morino TSO-MON product. CICS and IMS transaction times can be obtained from the CICS CMF and IMS DCMON facilities respectively. If all service levels are being met, then the analysis can stop and we can go on to other projects. There are always other projects.

To carry the example further, however, suppose that a critical batch job is running too slowly to meet our service goals. Suppose further that analysis [10] of the relevant SMF data has ruled out delays arising from device allocation, contention for possession of datasets, intensive demands on CPU or memory, and other non-IO waits. We are facing an IO tuning problem.

Now we need data at a lower level of detail. Which entities (devices and files) are the source of how much delay? Probably the easiest route to the answers goes through the SMF Type 30 (Common Address Space Work) record. SMF can be told to generate these records at the end of each job step, and even at stated intervals within a step. Each record contains dozens of measurement fields. We are interested in the EXCP section of the record, holding one set of fields for each file used during each step of a job. The fields include the DDname (used by the running program to reference a specific file), the device accessed, and the number of blocks transferred between file and memory since the file was opened (not since the start of the interval).

So far we know which devices are heavily accessed by the job, but we don't know the nature of the delays. Connect time is only part of the story. We need the total of each type of delay (connect, disconnect, pending, and IOS) if we are to locate the tuning point with the most leverage. The RMF Device Activity Report for the same time interval gives the average delay suffered per IO event for each device in the system. We estimate the job's delays for each file by multiplying the recorded block count times each average delay for the device. The result of that arithmetic will be a matrix of files against delays. We need only scan the matrix to find the largest number, and we can start to think about what to do to reduce that type of delay on that device.

Once we have located this bottleneck in the IO subsystem, what are the tuning options? The first to consider is IO reduction. The systems programmer can take a few actions that will reduce activity to system datasets. The MVS/XA LLA subsystem eliminates traffic against load library directories. JES2 job journalling can be turned off unless the installation uses MVS Checkpoint/Restart. Unused SMF records can be suppressed, as can trace activity of all types. The presence of a performance database, by the way, greatly reduces tape and disk activity by the performance and capacity management staff who are usually among the system's most intensive users. There are a number of software products which use CPU/memory/IO trade-offs to cut back references to load libraries, catalogs, etc.

MVS has an option that interfaces standard AMs to the paging subsystem so that datasets are created and manipulated on "virtual devices" with "virtual IO" (VIO). Because the operating system has control of VIO's buffers and real block sizes, VIO can exploit the "fast-path" code used for MVS paging. If the system is not already paging-

Table 6. Effects of Tuning Actions - MVS/370

Action	Effects
Repair the equipment	R P C D S
Reorder data within dataset	D S
Reorganize data on volume	D S
Reorganize data across volumes	R P C S
Rearrange device configuration	R P C S
Change hardware technology	P C D S
Tune buffers	Q
Tune applications	Q
Reschedule workload	Q B R P C D S
Convert RESERVEs	R P S
Prioritize I/O servicing	(S)

Legend:
- Q IOS queue delay
- B IOS queue when device is busy
- R HOS or external RESERVE delay
- P Delay with device "not busy"
- C Delay for Control Unit Busy
- D Delay for Device busy
- S Average service time

Table 7. Effects of Tuning Actions - MVS/XA

Action	Effects
Repair the equipment	P D C S
Reorder data within dataset	D C S
Reorganize data on volume	D S
Reorganize data across volumes	P S
Rearrange device configuration	P S
Change hardware technology	P D C S
Tune buffers	I
Tune applications	I U
Reschedule workload	I P D C U S
Convert RESERVEs	P U S
Prioritize I/O servicing	(S)

Legend:
- I IOS queue delay
- P Waiting for path
- D Waiting for mechanical motion
- C Protocol or data transfer time
- U Device RESERVEd but idle
- S Device active time

constrained, VIO offers performance advantages over real IO for small sequential datasets. There can be a large benefit/effort payoff in changing logon PROCs to use VIO datasets for EDIT recovery datasets, for instance. In some environments, users have incentives to convert their datasets to VIO, but beware the job that allocates VIO space for sort work areas!

Physical IO to sequential and VSAM files can usually be reduced at the expense of memory. Larger blocksizes mean the same number of blocks can be sequentially scanned with fewer physical IOs. Consider the timing constants shown in Table 4. If a file of 80-byte records is stored in 80-byte blocks on a 3350-type device, the average delay per GET is 19 msec (assuming minimum SEEKs). If the same file is reblocked to "half-track" (2 9440-byte blocks per track), each physical IO reads in 118 records at an average delay of 0.22 msec, a reduction by a factor of 86. Furthermore, because each block must be bounded by an unused space, the higher blocking fits 3.7 times as much usable data onto each track.

Similar logic applies to VSAM sequential access. VSAM buffer allocations default to one data buffer and two index buffers. If the file is to be accessed sequentially, substantial benefits can be realized by setting the number of data buffers, BUFND, to 10 or more (depending on details of CI size and processing per record).

Non-sequential access is quite another matter, because excess blocksize means excess data movement for no gain. Notice, though, that absolutely random access is surprisingly rare. Patchwise random access is much more widespread, and modest increases in blocksize can often result in a measurable increase in the number of hits per block read in.

Non-sequential access to a VSAM dataset can be sped up through IO reduction of another sort. There may not be much opportunity to cut back on physical IO of data, but the index is neither as volatile nor as bulky as the data. Keeping more index blocks in memory means that fewer will have to be read in a second time. The index buffer count is controlled by the file's BUFNI parameter, defaulted to 2. For keyed access, this value should be at least as large as (n + 1), where "n" is the number of index levels, given in the VSAM catalog listing.

IO reduction is not a panacea, however. It is a general solution that may not fit our problem. Rather than pursue at length the cause and cure of each sort of delay, consider the arrays set out in Tables 6 and 7. Here we see the actions that can be taken during storage tuning, and the type of delay that each will affect (in terms of the relevant RMF descriptor and the corresponding field name in the SMF Type 74 (Device Activity) record for that operating system.

A few words about the action list. The effect of equipment repair will depend, of course, on what part of the path is broken. In our experience, the weakest link is the recording surface and the actuator, both of which have the greatest effect on the "device busy" or "connect time" indicators. However, don't lose sight of the secondary effects that can arise. Error correction activity can dominate a path to the exclusion of all other processing, so a repair of one device can benefit the entire string and others as well.

"Reordering data" applies mostly to pseudo-random files which are accessed sequentially but which contain some data that is much more popular than others. A program load library is a prime example. If the most active programs are loaded physically adjacent to the directory, then head motion between directory and program is minimized.

IO prioritization is taking from Peter to pay Paul. During the IO life-cycle, IOS must be notified when the data movement has completed. This notification takes the form of an "interrupt", in which the current task is stopped so that system control may be transferred to IOS. If several IOs complete simultaneously, IOS must choose which to service. Two disciplines are supported: first-come-first-served and priority. In the latter, each IO interrupt inherits the relative priority of the task that generated the request. Prioritization will help some tasks but hurt others. It is a valuable tool, but requires that the installation make decisions about which class of work to favor.

SUMMARY

From this small survey of operating systems, we can draw several conclusions about the tuning process in general. First, tuning effectiveness depends on the number of dials and the number of knobs. We can't tune what we can't measure, and we can't tune what we can't adjust. Second, tuning responsiveness depends upon ready availability of data and ease of making changes. If measurements can't be made in real time, and adjustments can't be made in real time, neither can improvements.

It is important to treat the tuning process as a cycle, driven by comparison of current status to externally-set service goals. Without constant comparison between the real and the ideal, tuning becomes a pasttime rather than a purposeful activity.

General system tuning works well with a ROT approach, but specific performance problems should be addressed by looking for the significant sources of delay. Averages often hide more than they show, and experience shows that the extreme case is often the (politically or financially) most important one.

Finally, there is no substitute for deep knowledge of the system being tuned. Intuition is a powerful aid in the tuning process, but intuition without knowledge is metaphysics.

REFERENCES and NOTES

[1] - The word "Apple" and the phrase "Apple Pascal" are registered trademarks of Apple Computer, Inc. "IBM" is a registered trademark of the IBM Corporation. The phrase "UCSD Pascal" is a registered trademark of the Regents of the University of California. The word "UNIX" is a registered trademark of Bell Laboratories and its successors. The words "MS-DOS" and "Microsoft" are registered trademarks of the Microsoft Corporation.

[2] - The Apple hardware is described in "Apple II Reference Manual", Cupertino CA: Apple Computer, Inc. (1979). A more detailed exposition of the Apple II I/O subsystem can be found in Worth, D., and Lechner, P., "Beneath Apple DOS", Reseda CA: Quality Software (1981).

[3] - The Apple implementation of the UCSD dialect of Pascal is described in "Apple Pascal Language Reference Manual", Cupertino CA: Apple Computer, Inc. (1980). The operating system is covered in a separate manual, "Apple Pascal Operating System Reference Manual", Cupertino CA: Apple Computer, Inc. (1980).

[4] - K. Jensen and N. Wirth, "Pascal User Manual and Report" (2nd ed.), New York NY: Springer-Verlag (1978).

[5] - The IBM-distributed version of MS-DOS (also known as "PC-DOS") is described in "Disk Operating System Reference", Boca Raton FL: IBM Corporation, Personal Computer (1985). Many of the more arcane aspects of MS-DOS are discussed by Charles Petzold and others in the "PC Lab Notes" and "PC Tutor" articles that appear regularly in "PC Magazine".

[6] - These timing constants have not been distributed by the vendor. The values appearing here were

determined by measurements on the author's equipment and may not apply to other specific hardware items.

[7] - The UNIX literature is in a particularly chaotic state, mostly due to the large number of vendors producing more-or-less compatible versions of the base operating system. A suitable lead reference is Deitel, H. M., "An Introduction to Operating Systems" (rev. ed.), Reading MA: Addison-Wesley (1984), pp 103ff. Material specific to UNIX V can be found in vendor manuals such as the the set published by AT&T under the collective title "AT&T 3B2 Computer UNIX System V" (1985).

[8] - See the references cited in [9], especially items 1-5 and 7.

[9] - Olcott, R., "Online Storage Management in the Multi-processor Environment", Proc. CMG '85, 791-812 (1985).

[10] - Performance analysis of MVS workloads is discussed in Merrill, H. W., "Merrill's Expanded Guide to Computer Performance Evaluation Using the SAS System", Cary NC: The SAS Institute (1984).

ONLINE DATA BASE VS. RAW SYSTEM DATA

Robert H. Denis
Transamerica Information Services

ABSTRACT

For someone who is new to either performance evaluation or ad hoc management reporting, the collection of the necessary system data can be intimidating. This is espcecially true if the analyst has only a basic understanding of the programming language which is used to extract this data. The purpose of this paper is to show how a well structured online data base can help a new analyst develop meaningful and useful reports even with a limited comprehension of the programming language and the raw system data. As a result, the analyst will be able to spend his time performing analysis, rather than sifting through data. For illustrative purposes, the MICS Data Base and SAS language will be used.

INTRODUCTION

Every performance analyst or programmer has to go through the awkward stages of being new in that particular field. If SAS was the main language used, the analyst was probably sent to a basic class and given manuals to look through. He would obtain a good understanding of these basic skills and be of the opinion that he could handle most projects that came his way. However, he learns that the System Management Facility (SMF) is the main source from which most of the reporting will be done. The first examination of the layout of this data reveals that the SMF record is not the simple straightforward record that he had worked with previously. This paper will illustrate how this situation can be handled so that the analyst can respond to an SMF request even with just a basic understanding of the SAS language. The answer is the online data base. Before the data base is described, it is necessary to understand the general format of the SMF record.

THE RAW SMF RECORD

The SMF record is the means by which several computer functions can be monitored. Performance analysts and capacity planners use this record to measure such functions as throughput and resource usage. While SMF records are not overly complicated, they do not follow a fixed format. That is, the data is not in a format where each line contains fields in exactly the same columns for every record. The SMF record is a variable length record. It always starts with a header which contains among other things the numeric identification of the record. For example a type 5 record is created at the termination of a job in the system. Next there is a header unique to the record type which contains such information as entry length, number of entries on the record, and a record subtype identification if applicable. The remaining part of the record contains the data unique to that record type. This part of the record is an entry. There is usually more than one entry on a record. At the end of one entry a new entry starts on the same line. It is at this point that the analyst realizes that he is not dealing with a simple basic record. He must be able to hold data so that the entire record can be read. Then the offset of the fields in each entry must be considered because of the variability of the entry length. As a result the column numbers are not constant. In addition the numeric fields in the SMF record are not just plain numbers. Some of these fields are in integer binary and packed decimal formats. Time fields are in such formats as integer binary, time, and smfstamp formats. Some time measurements are made not in seconds, but in microseconds (tenths of seconds). Often these time fields must be manipulated requiring conversions to SAS time and back again when printed out. A well organized online data base can simplify this data extraction process significantly. Morino Associates' MICS Data Base shall be used to illustrate this point.

ONLINE DATA BASE

In order to have a workable online data base for SMF records, it must be able to provide the same data while simplifying the coding required. The MICS Data Base takes a giant step toward this goal. The logic needed to access the raw records is built into the source code. Each field of data is given its own unique name. Since each field name is unique, the analyst does not need to know in what columns the piece of data resides. The data base does that for him. The data is not in several different formats. All fields are either character, numeric, SAS time, or SAS timestamp. These formats are also part of the source code. If these fields are going to be printed on a report, no FORMAT statement is needed as the data base automatically prints them using the proper format.

Now that we have all of these handy fields, the question is how can they be organized for easy access. Related pieces of data are put into information areas such as Batch Utilization, Hardware Utilization, and so on. Each information area contains files. These files are given their own unique names. It is in these files that the various fields reside. Data from several SMF record types can be stored in the same database file.

In addition to data taken directly from the raw SMF records, additional fields can be created which contain computations from other fields on the data base. These fields can hold minimum values, maximum values, averages, percentages, and so on. This extra computed data can assist in reducing SAS code even further.

Another coding time saver is to have the data summarized by convenient units of time. This can be done in timespans such as DAYS, WEEKS, MONTHS, and YEARS. Within each timespan the data can be summarized to a level of time such as hour, day, week, month, or work shift. For this latter option MICS has a field called ZONE which allows the specification of unit of time other than hour, day, week, and so on. This time measurement could be work shift, peak and offpeak periods, or any period of time desired. A timespan can also be used to represent unsummarized data. This data would have the same granularity as the raw record. For example, a raw RMF record is created every fifteen minutes. In the data base an observation would be available which covers this fifteen minute period. MICS does this in a timespan called the DETAIL timespan. This is a daily timespan, but the difference between DETAIL and DAYS is that DETAIL data is unsummarized and DAYS data is summarized by hour. So now we have all this data nicely organized into fields, files, and timespans, but how does the analyst know what each field on the data base means? This is accomplished with a data dictionary.

The data dictionary gives an explanation of exactly what each piece of data represents. In the case of the MICS Data Base various pieces of information are

included. Of course a clear definition of the field should be the first item mentioned. The derivation of this field from the raw SMF record is included either as part of the definition or as a separate item. Next is the SAS format of the data. Then an explanation of how this field is handled in summarization is given. As an aid to the analyst a brief explanation of how this field can be used is present. Finally, a list of related fields that give similar information can also be included. The data dictionary can also have explanations on general topics which might be of interest to the analyst. These topics can be things like the effect of a new release on data collected, how various measurements are done, or anything related to the data base which can be explained fairly briefly. A data base has now been created which has all of the data well organized and a data dictionary to provide explanations of what the fields are. To make it easier to obtain data for a certain period of time MICS uses the concept of aging. For example, using the DAYS timespan, the first days data becomes cycle 01. The next example, using the DAYS timespan, the first days data becomes cycle 01. The next run moves this data into cycle 02 and the new day's data is put in cycle 01. On the third day, cycle 02 becomes cycle 03, 01 becomes 02, and the new data goes into cycle 01. This process continues each day. Let's say that 14 cycles are being kept online. The data in the 14th cycle will drop off the data base on the next run. Tapes can be created to supplement the online data base. These tapes allow longer retention of data without taking up disk space. MICS also provides a cycle 00 for WEEKS, MONTHS, and YEARS. This allows for either week-to-date, month-to-date, or year-to-date data to be kept. Now that we have a well organized data base it is necessary to take an additional step which is to minimize Job Control Language.

A Job Control Language Library should be created to hold various procedures needed for execution. Having these procedures as part of the data base reduces the amount of JCL required for the analyst to have in his application. In the case of MICS several procedures exist for various purposes. For brevity the MICSSHR procedure shall be used as an example since this is the procedure most commonly used by analysts. MICSSHR allows read only access to a single MICS data base unit. A unit is simply an organization of related data. For example, SMF data can be one unit, IMS data another unit, and so forth. Among other data sets, this procedure contains the data set names of each timespan with DDnames corresponding to the name of the desired timespan. Thus when this procedure is specified in the EXEC statement of the JCL, all of the JCL is generated.

For a simple example let's assume that a request has been made to list all executions of job A1234567 for the previous day. This report is to include the system and time the job ran. Upon looking at the MICS files, it is discovered that the data we need is part of a file called BATJOB. Since this file is part of the SMF data base unit, the single character ID for that unit is the letter S. This ID varies from one installation to another. Since the data requested is for the previous day we want cycle 01 on the DETAIL timespan. This file does not have a DAYS timespan and if it did we would need unsummarized data anyway. The specific fields required would be JOB for the jobname, SYSID for the system name, STARTTS for the time the job started running, and ENDTS for the time the job ended. Since a data base unit is being accessed the MICSSHR procedure shall be used. The code needed to do this is as follows:

```
// job card
//STEP1 EXEC MICSSHRS
//SYSIN DD *
DATA JOBS(KEEP=JOB SYSID STARTTS ENDTS);
  SET DETAIL.BATJOB01;
  IF JOB='A1234567';
PROC PRINT DATA=JOBS;
  ID SYSID;
  VAR STARTTS ENDTS;
TITLE EXECUTIONS OF JOB A1234567;
```

The data base unit identifier "S" is added to the end of the MICSSHR procedure name in the JCL. Note the absence of an INPUT statement. MICS is accessed using a SET statement. The first part of the SET statement indicates the timespan desired. DETAIL is a DDname in the MICSSHR procedure. Then after the period the name of the file, BATJOB, is indicated. The next two positions indicate the desired cycle. Since the previous day's data is needed, cycle 01 is given. A KEEP statement shows what fields are desired from the BATJOB file. Remember that no formats are needed since MICS handles that automatically. An IF statement is present because only those observations with jobname A1234567 are required. Next a report is created with a PROC PRINT. In the PROC PRINT illustrated the observation line numbers are being suppressed and the system id is the left most column. The ID statement performs these two functions. The VAR statement then indicates that the next two fields to be shown on the report are STARTTS and ENDTS in that order. Because these two values are MICS fields there is no need to have a FORMAT statement. They will automatically be printed using the DATETIME format. Finally a TITLE statement for the name of the report concludes the program.

If a new analyst had to write this same program directly from the raw SMF record, he would need to find the location of each field. As mentioned earlier, this is not as straightforward in SMF. The location of a field is usually the result of a calculation rather than an actual column number. He would need to be aware of the format of each piece of data as well. In other words, the analyst would be spending extra time figuring out what the INPUT statement should be. The program would also need FORMAT statements for the time fields so they are printed in a readable format. In MICS all that would be needed would be to look up the proper file and find the names of the needed fields without worrying about where the field is located. In a more involved program calculations may be needed on some time fields. These fields are already in a SASTIME format in MICS. The conversion has already been done in the source code. Another common SAS statement may isolate parts of DATETIME field such as month, day, and so on. MICS has fields such as MONTH, DAY, YEAR, HOUR, etc. which makes most of these statements unnecessary. As a result time is freed up for doing analysis rather than sifting through data and extra SAS code. Unfortunately the many benefits provided by an online data base do not come free. The next section discusses both the advantages and disadvantages of the data base approach.

ADVANTAGES/DISADVANTAGES

In this section the main advantages and disadvantages of the online data base are summarized. The main advantages are as follows:

1. The new analyst is able to produce practical reports sooner with a limited amount of SAS knowledge due to the simpler record format.

2. A large volume of data is available online, with tapes used for long term data.

3. Several fields are summarized in higher timespans. As a result fewer observations are needed to retrieve data covering various periods of time. To illustrate, four RMF records would be needed to cover an hour. In MICS only one record would be required in the DAYS timespan. Summarization in this timespan is done to the hour.

4. More than one type of data is available on a single data base. For example, SMF and IMS data can be collected in one program by using a procedure that provides read only access to more than one data base unit.

As good as the online data base is, it does have a couple disadvantages which are as follows:

1. The data base is not available during updating. In other words programs cannot be executed against the data base while data is being reformatted from the raw record.

2. A data base of this type tends to use a large amount of disk space. In the case of MICS not all timespans are activated for all files. Various value judgements were required to best produce a data base which is practical while using a minimum amount of space. Disk space will vary depending on the amount of data the user desires to keep online.

3. People and machine resources are required to establish, maintain, monitor, and update the data base itself and the data base system.

4. Fields and summarizations may not always be in the desired form.

5. By using data that is "one step removed" from the originally recorded data, one always runs the risk of making mistakes of interpretation or misunderstanding.

CONCLUSION

Any new analyst faces the responsibility of bringing himself up to speed as efficiently and quickly as he can. This task is done through training and on the job experience. If part of his duties is the accessing of SMF data, then an online data base can make this learning process go a little faster. In using the MICS Data Base as an example, it has been shown how a well organized collection of SAS files can simplify SMF data extraction. As a result the new analyst feels a little less awkward and gains confidence that he can do his new job well.

METHODOLOGY TO DEVELOP A TOOL FOR MONITORING CPU USAGE BY APPLICATIONS RUNNING UNDER VM

Robert D. Deese
Hossein H. Ghannad
IBM Corp.

ABSTRACT

In CMG 86 at Los Vegas, Nevada a need for a tool that would report different applications CPU usage running under VM was mentioned more than once. This is very important information for a capacity planner who has to determine in a timely fashion whether a larger processor is needed. A given VM installation may run several different applications. The CPU resource requirement of these applications are usually different and when a user growth projection is made, they are made for each and every application. The combination of knowing the average CPU resource requirement for each application and the projected usage growth for that application allows the capacity planner to accurately project the size of the CPU to be obtained.

In this paper we will describe a step by step methodology to build a simple tool using existing VM capabilities (REXX and CP internal timer facility) to monitor CPU application usage by one or more users.

1. INTRODUCTION

One of the most important function that a capacity planner performs is to order and procure a higher capacity central processing unit (CPU) before the existing one becomes over loaded. This is important for two reasons. First, it usually takes between three to twelve months from the time the determination for a larger processor is made and the time the processor is actually installed. Secondly, as the utilization of any computer resource, including CPU, approaches its threshold, usually 90-95 percent for CPU, the response time passes the knee (or elbow) of the throughput/response time curve (see figure 1) and becomes excessively large. This is not a desirable situation to be in for a capacity planner unless he/she likes to spend a lot of time, usually on the telephone, explaining long response times to users who in many cases have to interface with customers who may be even less patient.

One of the ways capacity planning is usually done, as far as CPU is concerned, is by determining CPU consumption for an 'average' user for an 'average' session. The projection of increase in the number of users is usually obtained from other departments in an organization. Simple arithmetic yields the amount of more processing power which would be needed if the increase in the number of users projection is realized.

One of the problems that many capacity planners run into is determining resource consumption, in this case CPU, by the 'average' user. For some systems this is not difficult because all users of that system interface with the same application and basically use the system for the same type of activities.

VM/CMS, is an interactive system which is used for many purposes such as program development, numerically intensive computing, document development, office management system by the same or different users. The CPU requirements of these activities may be considerably different. A good example of this is the difference between CPU resources requirements of a physicist, who mostly runs numerically intensive routines that are precompiled and stored on a system, versus an executive or secretary that may deal mostly with office communication and scheduling. If the projected growth in user population will have a different distribution than the present one or the character of the user population is to change, that is, same group of users using the system differently than before, then the capacity planner needs to know the CPU requirement of an 'average' user for a given type of activity or how much CPU is consumed by an application per user. In the sections that follow we introduce a methodology which uses existing VM/CMS capabilities and functions provided by the system to answer the questions needed by capacity planner to make a more accurate projection of the size and time of CPU upgrade.

It is also our intention to show that this tool can be easily built within ones own shop with little or no additional skills. As is true with any programming tool, there is more than one way of obtaining the same solution, including the use of other programming language. We used the Restructured Extended Executor(REXX) language, which is processed by the System Product Interpreter, and the Control Program (CP) internal timer facility. It is the authors opinion that learning the REXX language is easier or as easy as learning any other 'high level' programming language. The use of timer facility is very trivial and is matter of executing a system provided command (Query TIME). The objective is to implement a monitoring tool which will be virtually transparent to the user. This tool will calculate the total CPU time (virtual and overhead) and the virtual CPU time used per application and also keep track of the total elapsed time while using these applications. A running tally must be kept which can be either directly used or reduced and used by the capacity planner for his or her user growth projections.

2. SELECTING USERS

The number and type of users to be selected for monitoring depends on several factors. First, the capacity planner needs to obtain the user and/or application usage growth projection from other departments based on the establishments future activities and growth. Secondly, the capacity planner needs to determine the type of users and their computer system activities. For example, there may be a projection of thirty percent increase in office management system usage, ten percent in data base management application usage, and three percent in numerically intensive computing type usage.

The actual number of users to be monitored depends on the variability of resource usage by the users. This may be statistically determined with a high confidence level if statistics (means and variance) for the processor usage is available. The rule of thumb is the larger the variation, the more users should be selected. If there are no statistics available, a few users of the applications under study may have to be monitored before one can determine how many users of each application should be included in the study to project the future CPU capacity need. We should point out that sometimes additional sessions from the same user may be substituted for more users. This can be done if users engage in different types of activities from day to day.

Once the user selection for monitoring has been made and the number of sessions determined, the tool and an introduction to it should be sent to the selected users. The introduction should include a brief description of the tools purpose, similar to an abbreviated version of our introduction, and a brief description of the tool and its function.

Once the tool has been stored on the users disk, they simply include its invocation in their user profile. From there the tool will run transparently.

3. THE TOOL DEVELOPMENT METHODOLOGY

The first step to collecting the needed data is to create a CMS input file. This is easily achieved in REXX by using the System Product Editor (XEDIT). Before the file is actually created, two commands are needed, which are put in the command queue. These are the commands to write the current data to the file and to store the file on the users disk for further update. The data is used for two purposes. First, is for tracking and recognition of the trace, and second, to enable storing the file on the disk (the system will not store an empty file on the disk). Once these commands are in the queue, the file is created by issuing an XEDIT command in the REXX program. The date inserted, the file is stored on the users disk for further data input over the monitoring period.

Once the above action is completed, the user is prompted with a ready message to start his work with the tool running transparently. From this point until exiting the tool, commands entered will be read into a template (a special variable to store strings). A 'while' loop is used to read every command that issued by the user. If the command issued by the user is not the 'LOGoff' command, the code block that holds the bulk of the program is entered via a subroutine call.

This subroutine will issue a query of the CP internal timer facility and assign the returned information to a variable called STARTCOMMAND. The command who was stored in the template is then issued for execution by the system. The execution of the command is carried out as usual without any interference by the tool. Upon completion of the command, the CP internal timer facility is again queried and the returned information assigned to another variable called FINISHCOMMAND.

To convert the times which are in string form to real time in seconds three function calls are made to two different functions. The Querying of the time supplies the STARTCOMMAND and FINISHCOMMAND variables with time and date information, from which, the tool uses the following: the actual clock time spent in the current terminal session, the virtual processor time and the total processor time used in the current session terminal. The actual clock times were in hours, minutes, and seconds which were converted to seconds in the functions, before total elapsed times were calculated. The other two times (VTIME, TTIME) are given in minutes and seconds. Therefore, we used a different function to convert and calculate the VTIME and TTIME. The functions started by parsing the string stored in STARTCOMMAND variable to isolate the processor times or actual clock time. This substring is then converted into seconds and put into STARTTIME variable (the actual clock time also contains a hour variable which also needs additional conversion). Upon completion of the command stored in the template, this process is repeated with the FINISHCOMMAND variable by calling the same functions. This time the functions return the converted values in FINISHTIME variable. Then STARTTIME variable is simply subtracted from the FINISHTIME variable to generate the processing and/or elapsed time per command.

Many commands are truncated to give a shorted version of the command word for ease of use. Many users also do this for their own use in a synonym file. The truncated commands and synonyms are compared with the full command word and all truncated commands and synonyms are then converted to the full command word. This is done to make the reduction of the data and calculating total times per session for a given command easier.

With the needed times calculated and the command words in suitable form, they are gathered together and formatted into one variable. A command which allows for I/O operations to VM/CMS files is now issued to write the formatted variable containing the required data to the monitoring file.

The user is now prompted for their next command. If the entered command is not the 'LOGoff' command, the loop is reentered and the above process repeated. However, if the user is logging off their machine, the loop and the program is exited. The monitoring file is then sent to the capacity planner or other person responsible for monitoring the data. The monitoring file is erased from the user minidisk and the user is logged off.

4. DATA REDUCTION

As we mentioned earlier, the monitor program writes the command, the total processing times (VTIME and TTIME), and elapsed time for each command into a CMS file. We wrote a simple PASCAL program to reduce the data. The program first reads the command, whose total processing time during the session is desired, from a CMS file. This file is created using XEDIT and includes a list of one or more commands. Then the reduction program searches the monitor file for all the occurrences of that command and adds the three above mentioned times together and writes the command and totals into a third file for later inspection. We used an appended function of CMS to create one file from several sessions of the same user or from several users whenever we wanted to look at accumulative CPU consumption of each command for more than one user or session.

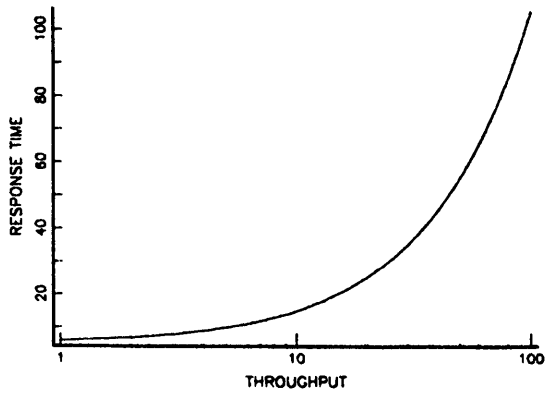
5. SUMMARY

In this tutorial paper we restated the need to develop a tool to monitor and report the CPU consumption of applications running under VM/CMS. We introduced a methodology to develop such a tool using existing VM/CMS facilities. We also presented a high level algorithm which we coded in REXX, and described a simple data reduction program to provide the results for analysis or further reduction. It is the authors opinion that a tool built using our description would be of great help to capacity planners of VM systems in determining their future processing needs.

HIGHLEVEL DESIGN OF OUR ALGORITHM:

```
Create a Monitor File
Enter Desired Command or Application
While the Entered Command is Not LOGoff DO
  Call a Subroutine to:
    Query the Time
    Execute the Command or Application
    Query the Time
    Call a Function to Calculate Total CPU Usage
    Call a Function to Calculate Virtual CPU Usage
    Call a Function to Calculate Total Elapsed Time
    Format the Collected Data
    Write the Formatted Data to the Monitoring File
    Return
  Enter Next Command or Application
End or Repeat the Loop
Send the Monitor File to Persons responsible for
  Data Collection
Erase the Monitor File
Log off
Exit
```

FIGURE 1– A TYPICAL RESPONSE TIME/THROUGHPUT CURVE



CICS PERFORMANCE AND WORKLOAD REPORTS

Chuck Comstock
Hallmark Cards, Inc.

ABSTRACT

Reporting the "health" of CICS systems was once a problem of lack of sufficient performance data. In today's world of RMF, CMF, TMON, and other performance monitors, the converse is more often the case. Reducing hundreds of thousands, or perhaps even millions of detail performance records into meaningful performance and workload reports is a significant challenge.

This paper will examine some of the more important performance and workload indicators and appropriate statistical measures, summary levels and trend analysis, and issues that should be kept in mind when creating or analyzing CICS performance and workload reports.

Introduction

Today there is a profusion of CICS information available from a variety of monitors and data collectors. Reducing this mass of data to to something meaningful and useful requires some understanding of what is available and what is needed. Regardless of the specific performance monitor being used, however, two broad categories of system metrics should be collected and reported: performance data and workload data.

Performance measures indicate how well the system satisfied requests for service. Performance indicators consist primarily of response time, or how long before a request was completed, and throughput, or how many requests were processed. Unfortunately there are several different ways of reporting these metrics. Considerations will be covered in more detail in the sections on Simple Descriptive Statistics and Summary Levels and Trend Reports.

Workload statistics show the demand for system service. Some workload measures are: transaction input rates, CPU utilization, data access request rates, and memory requirements.

Of course some measures fall into both categories. Transaction rates are primarily a measure of workload but, depending on availability of resources, they can also be a measure of performance.

Simple Descriptive Statistics

Simple descriptive statistics should be just that: simple, descriptive measures of performance and workload. Most of the classical statistical measures are useful when reporting one or another of the performance and workload indicators. Some forms are more appropriate with certain metrics and considerations will be covered with each statistic below.

Some of the most frequently used statistics and metrics are:

- Averages
- Sums (totals)
- Minimums/maximums (range)
- Percentages/percentiles
- Rates

Averages

Perhaps the most frequently used and, therefore, abused statistic is the average. "Average" response time is often the performance analyst's cop-out, particularly when it represents goodness.

Our obsession with the average is probably due in large part to the fact that it is something easily related to. For example, most can easily tell the difference between half second response and five second response, or, as an old saying has it, if you notice response, it is too long.

The problems with averages that most of us have experienced fall into two categories: (1) outliers that ***skew*** the average up, e.g., the presence of a few very long running transactions, and (2) outliers that ***skew*** the average down. While the latter may usually be dismissed, the former is hard to ignore.

One solution to the problem of a few outliers skewing an average is to use a ***percentile*** or ***percentage distribution*** approach. For example, instead of reporting average response time, express response in terms of a percent of the total completing in less than some arbitrary time, say, three seconds.

The disadvantage of the percentile approach is that it may be difficult to relate to. The obvious advantage is that a few extreme observations will not skew it. An excellent discussion of the percentile/distribution technique may be found in the *Morino Associates Technical Tips, Volume II.*

Another solution to the average skew problem is to discard, or at least ignore, the outliers. Since most long running transactions are doing "background" work anyway, their elimination from the reported response time is justified. Our view of response time is that what we should report is what the true ***interactive*** user is experiencing, not some non-terminal attached "batch" transaction.

One technique of determining if a transaction should be ignored when calculating the average is to set thresholds for resource consumptions. Transactions that exceed the thresholds become candidates for exclusion from the average response time calculation. A transaction that performs several thousand data base calls, for example, can hardly be expected to execute with subsecond response time, at least with most of today's software and hardware.

Fortunately, most performance monitors and post-processors allow some degree of control over the transactions used for reporting. Morino's MICS system, for example, provides a ***relative longevity*** exit (CICRLRT) that allows for ***typing*** CICS transactions based on any user developed algorithm. Transaction typing has existed in the TSO world for some time as a real-time prioritizing technique (are you listening, Hursley?), and while not available to us in real time, at least the concept is useful after the fact.

In reality it may be appropriate to report both averages and percentiles, understanding the limitations of each. If the average exceeds the bounds of good taste we at least can refer to the percentile to gauge what the appropriate level of crisis reaction should be.

The following are candidates for reporting as averages or percentage distributions:

- Response Time (Internal)
- Wait Time (VSAM, DBMS, Journal, Terminal)
- Suspend Time
- CPU Time

Sums (Totals)

Our delight in small numbers for response time is equalled only by our joy in large numbers for total transactions processed.

While a one-million transaction day may be impressive to the casual observer, to the performance and workload concerned individual this total is only interesting in so far as it relates to yesterday, or last week, or last month, or last year. Summary levels and historical trends will be discussed in more detail later; suffice to say that totals may be a useful measure of not only workload but also performance.

Some measures that may be reported as totals are:

- Transactions
- Data Access Requests (VSAM, DBMS, Journal, Terminal)
- Availability (Up Time/Down Time)

Minimums/Maximums

Minimums and maximums may be considered a high level view of variability, a measure easier to understand than standard deviation or coefficient of variation. In reality, it has been our experience that maximums seem to be more helpful than minimums and minimums can safely be ignored in most cases.

A maximum, taken along with an average, can be informative. If the variance between the two is large we know the possibility exists that the average has been skewed up to a significant degree. Also, particularly when analyzing several transactions, the presence of similar large maximums may be an indication of a common problem such as file string contention, enqueues, or even hardware problems.

The most useful indicators to report as maximums seem to be:

- Response Time (Or Wait Times)
- CPU Utilization

Rates and Percentages

Rates, such as transactions per second or data base calls per transaction, are useful because they take a sum or total and make it into a more easily understood number and make comparisons easier. For example, isn't it easier to relate to 4 transactions per second than 345,600 per day?

Likewise for percentages. Knowing that a CICS system consumed 28,800 seconds is not very revealing. Understanding that this is 80% of the system may raise a few eyebrows.

The following are good indicators to report as rates and percentages:

- Transactions Per Second (Minute/Hour)
- CPU Utilization
- Data Access Requests Per Second (VSAM, DBMS, Journal)
- Data Access Requests Per Transaction
- Page Ins Per Second
- Page Ins Per CPU Second
- CPU Per Data Access Request
- Terminal Characters Per Transaction

Summary Levels And Trend Reports

From detail, transaction by transaction reports to the annual summary, there are many different levels of summarization possible. Consider:

- Detail (1 observation per transaction)
- Interval (5-60 minutes)
- Hourly
- Shift (zone)
- Daily
- Weekly
- Monthly
- Quarterly
- Yearly

Each summary level is appropriate for different levels of management, from the performance analyst to the MIS Director or the executive board.

The sample report shown at the back of this paper is a daily report that reflects daily totals, some shift breakdowns, an hour by hour summary, and some trend reports in the form of daily averages and summaries.

It has been our experience that this report works well from middle management down to the performance analyst. The first page (daily summary and averages) is often the only page referenced. Anyone with experience using this report has a fair idea of what is "good" and what is "bad." If the first page reflects goodness, then that is it. On the other hand. . . .

When an element of evil creeps into the daily summary we would like to have as many answers as possible contained within the report without having to refer to other data or run additional reports.

The hourly reports provide some insight into the distribution of the "badness" and, if the problem is isolated to a small portion of the day, the approximate time of the difficulty. Of course, if the whole day looks bad, that tells us something as well.

Finally, the trend reports, charts or significant performance statistics by day for the last week or so, can reveal just how bad the situation has become and whether or not the problems have occurred recently.

Almost identical reports are also produced on a weekly and monthly basis.

Closing Comments

One of the significant performance software developments of the last 5-10 years has been that of the Performance Data Base (PDB) management systems such as Morino's MVS Integrated Control System (MICS), BGS Systems Info-Base, and Merrill's MXG.

These systems build data bases of performance data summarized to the individual user's needs (and availability of DASD to store it!).

Several of these systems use SAS Institute's SAS as the base language and data management system, or provide interfaces to SAS. The power and flexibility of SAS in conjunction with a PDB was a major factor in our ability to provide a variety of reports at various summarization levels.

While Artificial Intelligence and Expert System technology will address the issues discussed in this paper, there may always be a need for reports similar to those outlined here. The challenge will always be to provide the most amount of required information in the most meaningful manner in the least amount of space.

The author gratefully acknowledges the assistance and advice provided by Shawn Wikle, Joe Bell, Irwin Kraus, Nathan Walsh, and technical support personnel from Morino Associates.

D A I L Y C I C S A C T I V I T Y R E P O R T

PRODUCTION SYSTEM SUMMARY

CICS SYSTEM ID=KCP1 DATE=MONDAY, AUGUST 24, 1987

CICS SYSTEM ID	TIME ZONE	TRAN COUNT	AVG RESP	< 3 SEC	AVG RESP - S/M/L	PCT - S/M/L/X	UP TIME	DOWN TIME
KCP1	PRIME TIME	178,645	0.66	99 %	0.47 1.52 2.10	68 12 2 18	10:00:00	0:00:00
KCP1	NIGHTS	13,124	0.17	100 %	0.09 0.45 1.01	53 10 2 35	14:00:00	0:00:00
TOTAL		191,769					24:00:00	0:00:00

CICS SYSTEM ID=KCP2 DATE=MONDAY, AUGUST 24, 1987

CICS SYSTEM ID	TIME ZONE	TRAN COUNT	AVG RESP	< 3 SEC	AVG RESP - S/M/L	PCT - S/M/L/X	UP TIME	DOWN TIME
KCP2	PRIME TIME	73,698	0.43	99 %	0.32 0.66 2.84	74 13 2 11	10:00:00	0:00:00
KCP2	NIGHTS	962	0.17	100 %	0.13 0.18 2.24	66 15 1 19	14:00:00	0:00:00
TOTAL		74,660					24:00:00	0:00:00

CICS SYSTEM ID=KCP3 DATE=MONDAY, AUGUST 24, 1987

CICS SYSTEM ID	TIME ZONE	TRAN COUNT	AVG RESP	< 3 SEC	AVG RESP - S/M/L	PCT - S/M/L/X	UP TIME	DOWN TIME
KCP3	PRIME TIME	133,475	0.65	99 %	0.29 0.94 2.80	58 25 6 11	8:40:48	1:19:12
KCP3	NIGHTS	18,732	0.36	100 %	0.15 0.57 1.03	43 27 6 24	14:00:00	0:00:00
TOTAL		152,207					22:40:48	1:19:12

CICS SYSTEM ID=KCP4 DATE=MONDAY, AUGUST 24, 1987

CICS SYSTEM ID	TIME ZONE	TRAN COUNT	AVG RESP	< 3 SEC	AVG RESP - S/M/L	PCT - S/M/L/X	UP TIME	DOWN TIME
KCP4-KC/AWS	PRIME TIME	124,754	0.61	97 %	0.18 0.89 1.81	55 17 16 12	10:00:00	0:00:00
KCP4-KC/AWS	NIGHTS	9,337	0.38	100 %	0.06 0.43 1.13	52 15 21 13	14:00:00	0:00:00
TOTAL		134,091					24:00:00	1:19:12
		552,727						1:19:12

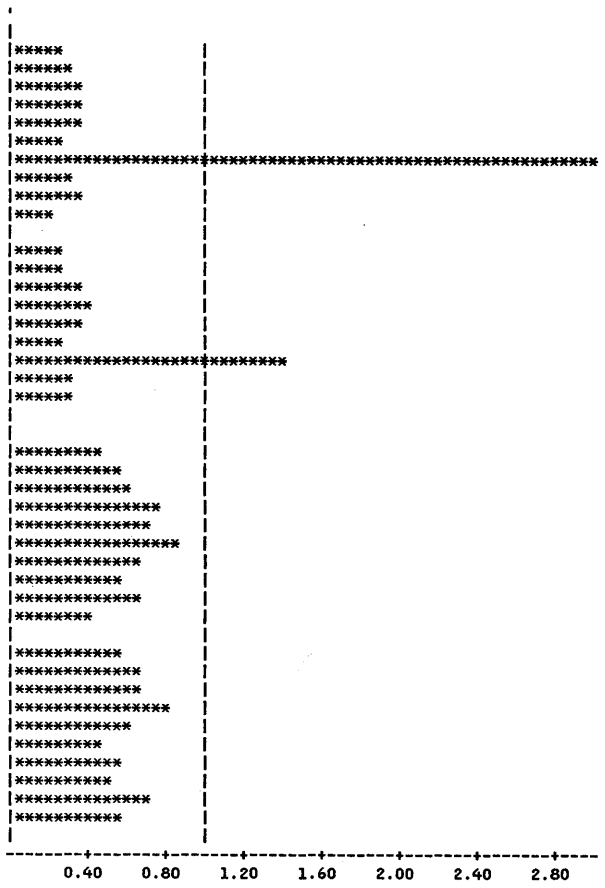
D A I L Y C I C S A C T I V I T Y R E P O R T

AVERAGE INTERNAL RESPONSE TIME (BY HOUR)

DATE=MONDAY, AUGUST 24, 1987

BAR CHART OF SECONDS

CICSID	HOUR	SECONDS
KCP1	7	0.23
	8	0.29
	9	0.34
	10	0.35
	11	0.33
	12	0.24
	13	3.00
	14	0.30
	15	0.35
	16	0.22
KCP2	7	0.26
	8	0.27
	9	0.37
	10	0.42
	11	0.34
	12	0.27
	13	1.42
	14	0.28
	15	0.29
	16	0.00
KCP3	7	0.47
	8	0.57
	9	0.60
	10	0.76
	11	0.72
	12	0.85
	13	0.67
	14	0.54
	15	0.67
	16	0.39
KCP4	7	0.55
	8	0.66
	9	0.65
	10	0.78
	11	0.59
	12	0.45
	13	0.55
	14	0.50
	15	0.69
	16	0.53



AVERAGE INTERNAL RESPONSE TIME (BY HOUR)

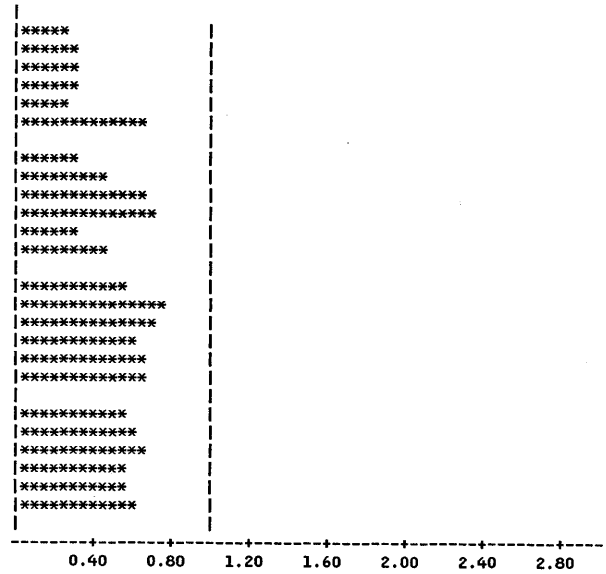
D A I L Y C I C S A C T I V I T Y R E P O R T

AVERAGE INTERNAL RESPONSE TIME (BY DAY)

TIME ZONE=PRIME TIME

BAR CHART OF SECONDS

CICSID	DATE	SECONDS
KCP1	MON, AUG 17, 87	0.27
	TUE, AUG 18, 87	0.31
	WED, AUG 19, 87	0.30
	THU, AUG 20, 87	0.28
	FRI, AUG 21, 87	0.25
	MON, AUG 24, 87	0.66
KCP2	MON, AUG 17, 87	0.32
	TUE, AUG 18, 87	0.45
	WED, AUG 19, 87	0.65
	THU, AUG 20, 87	0.71
	FRI, AUG 21, 87	0.30
	MON, AUG 24, 87	0.43
KCP3	MON, AUG 17, 87	0.53
	TUE, AUG 18, 87	0.77
	WED, AUG 19, 87	0.72
	THU, AUG 20, 87	0.59
	FRI, AUG 21, 87	0.67
	MON, AUG 24, 87	0.65
KCP4	MON, AUG 17, 87	0.56
	TUE, AUG 18, 87	0.60
	WED, AUG 19, 87	0.65
	THU, AUG 20, 87	0.56
	FRI, AUG 21, 87	0.55
	MON, AUG 24, 87	0.61



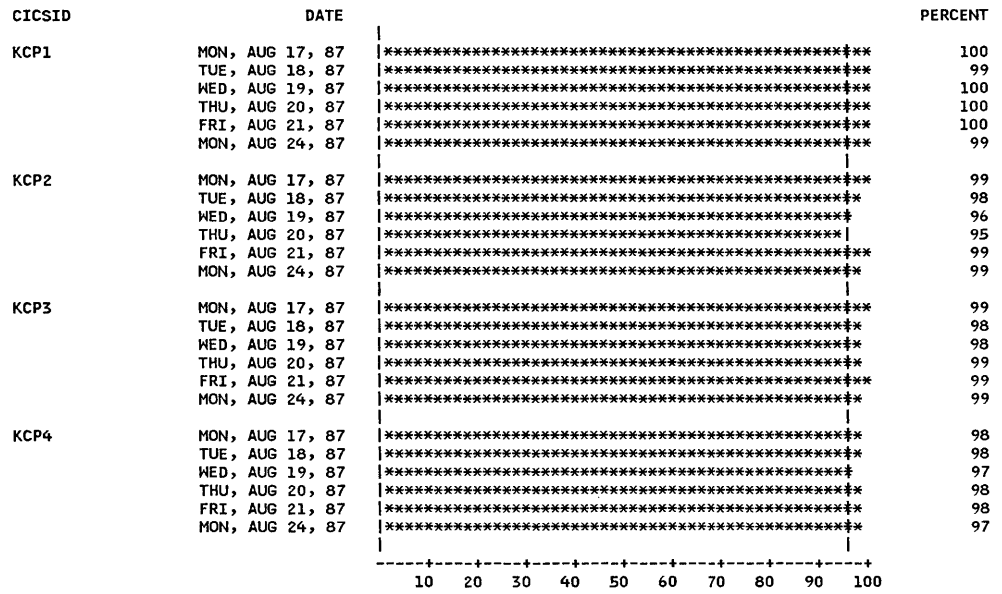
AVERAGE INTERNAL RESPONSE TIME (BY DAY)

D A I L Y C I C S A C T I V I T Y R E P O R T

TRANSACTION RESPONSE < 3 SECONDS (PERCENT BY DAY)

TIME ZONE=PRIME TIME

BAR CHART OF PERCENT

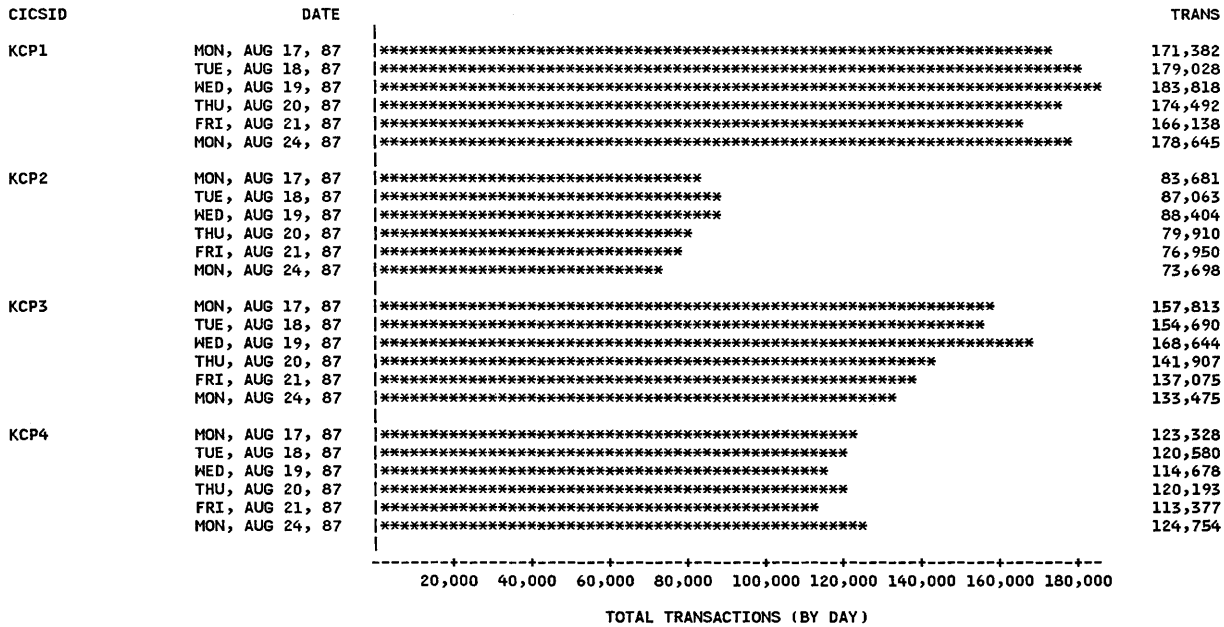

CICSID
DATE
PERCENT
KCP1
MON, AUG 17, 87 100
TUE, AUG 18, 87 99
WED, AUG 19, 87 100
THU, AUG 20, 87 100
FRI, AUG 21, 87 100
MON, AUG 24, 87 99
KCP2
MON, AUG 17, 87 99
TUE, AUG 18, 87 98
WED, AUG 19, 87 96
THU, AUG 20, 87 95
FRI, AUG 21, 87 99
MON, AUG 24, 87 99
KCP3
MON, AUG 17, 87 99
TUE, AUG 18, 87 98
WED, AUG 19, 87 98
THU, AUG 20, 87 99
FRI, AUG 21, 87 99
MON, AUG 24, 87 99
KCP4
MON, AUG 17, 87 98
TUE, AUG 18, 87 98
WED, AUG 19, 87 97
THU, AUG 20, 87 98
FRI, AUG 21, 87 98
MON, AUG 24, 87 97
10 20 30 40 50 60 70 80 90 100
PERCENT < 3 SECONDS (BY DAY)

D A I L Y C I C S A C T I V I T Y R E P O R T

TOTAL TRANSACTIONS (BY DAY)

TIME ZONE=PRIME TIME

BAR CHART OF TRANS



D A I L Y C I C S A C T I V I T Y R E P O R T

SYSTEM SUMMARY AND AVERAGES (BY DAY)

CICS SYSTEM ID=KCP1 TIME ZONE=PRIME TIME

DATE	RESP AVG	PCT <3S	TRANS	PAGEIN /SEC	PAGEIN /CPU	CPU %	VSAM /SEC	VSAM WAIT	DBMS /SEC	DBMS WAIT	JOUR /SEC	JOUR WAIT	CPU /IO	UP TIME	DOWN TIME
MON, AUG 17, 87	0.27	99.64	171,382	0.28	1.68	16.86	62	0.010	7	0.000	5	0.020	0.0017	10:00:00	0:00:00
TUE, AUG 18, 87	0.31	99.40	179,028	0.50	2.89	17.24	58	0.012	8	0.000	6	0.020	0.0018	10:00:00	0:00:00
WED, AUG 19, 87	0.30	99.55	183,818	0.46	2.64	17.59	58	0.012	10	0.000	6	0.021	0.0018	9:45:00	0:15:00
THU, AUG 20, 87	0.28	99.59	174,492	0.45	2.64	16.89	56	0.011	10	0.000	6	0.020	0.0018	10:00:00	0:00:00
FRI, AUG 21, 87	0.25	99.74	166,138	0.31	2.04	15.37	51	0.011	6	0.000	6	0.019	0.0018	10:00:00	0:00:00
MON, AUG 24, 87	0.66	99.10	178,645	0.87	4.72	18.40	66	0.010	9	0.000	7	0.021	0.0017	10:00:00	0:00:00

CICS SYSTEM ID=KCP2 TIME ZONE=PRIME TIME

DATE	RESP AVG	PCT <3S	TRANS	PAGEIN /SEC	PAGEIN /CPU	CPU %	VSAM /SEC	VSAM WAIT	DBMS /SEC	DBMS WAIT	JOUR /SEC	JOUR WAIT	CPU /IO	UP TIME	DOWN TIME
MON, AUG 17, 87	0.32	99.08	83,681	0.17	1.33	12.69	35	0.009	3	0.000	3	0.032	0.0026	10:00:00	0:00:00
TUE, AUG 18, 87	0.45	98.05	87,063	0.24	1.72	13.91	46	0.012	2	0.000	2	0.031	0.0023	10:00:00	0:00:00
WED, AUG 19, 87	0.65	96.48	88,404	0.29	1.97	14.68	42	0.022	2	0.000	2	0.030	0.0026	10:00:00	0:00:00
THU, AUG 20, 87	0.71	94.93	79,910	0.15	0.85	17.34	51	0.044	3	0.000	2	0.031	0.0026	9:17:24	0:42:37
FRI, AUG 21, 87	0.30	99.21	76,950	0.13	1.03	12.29	37	0.008	2	0.000	2	0.027	0.0026	10:00:00	0:00:00
MON, AUG 24, 87	0.43	98.80	73,698	0.31	2.57	12.11	34	0.010	2	0.000	2	0.038	0.0025	10:00:00	0:00:00

CICS SYSTEM ID=KCP3 TIME ZONE=PRIME TIME

DATE	RESP AVG	PCT <3S	TRANS	PAGEIN /SEC	PAGEIN /CPU	CPU %	VSAM /SEC	VSAM WAIT	DBMS /SEC	DBMS WAIT	JOUR /SEC	JOUR WAIT	CPU /IO	UP TIME	DOWN TIME
MON, AUG 17, 87	0.53	99.29	157,813	0.48	2.03	23.78	55	0.013	40	0.000	21	0.027	0.0013	10:00:00	0:00:01
TUE, AUG 18, 87	0.77	97.81	154,690	0.76	2.98	25.60	67	0.037	41	0.000	22	0.030	0.0013	10:00:00	0:00:00
WED, AUG 19, 87	0.72	97.84	168,644	0.65	2.23	29.27	71	0.015	53	0.000	26	0.032	0.0013	10:00:00	0:00:01
THU, AUG 20, 87	0.59	98.92	141,907	0.67	3.14	21.48	50	0.013	36	0.000	19	0.028	0.0013	10:00:00	0:00:01
FRI, AUG 21, 87	0.67	99.32	137,075	0.46	2.04	22.55	59	0.010	42	0.000	18	0.026	0.0012	10:00:00	0:00:00
MON, AUG 24, 87	0.65	98.77	133,475	0.11	0.46	24.56	62	0.011	42	0.000	20	0.028	0.0015	8:40:48	1:19:12

CICS SYSTEM ID=KCP4 TIME ZONE=PRIME TIME

DATE	RESP AVG	PCT <3S	TRANS	PAGEIN /SEC	PAGEIN /CPU	CPU %	VSAM /SEC	VSAM WAIT	DBMS /SEC	DBMS WAIT	JOUR /SEC	JOUR WAIT	CPU /IO	UP TIME	DOWN TIME
MON, AUG 17, 87	0.56	97.94	123,328	0.10	0.36	26.47	142	0.009	0	0.000	20	0.025	0.0013	10:00:00	0:00:00
TUE, AUG 18, 87	0.60	97.71	120,580	0.15	0.57	26.49	141	0.009	0	0.000	20	0.026	0.0012	10:00:00	0:00:00
WED, AUG 19, 87	0.65	96.74	114,678	0.15	0.58	26.57	140	0.009	0	0.000	21	0.028	0.0013	10:00:00	0:00:00
THU, AUG 20, 87	0.56	97.99	120,193	0.16	0.62	25.21	126	0.009	0	0.000	20	0.026	0.0013	10:00:00	0:00:00
FRI, AUG 21, 87	0.55	98.35	113,377	0.10	0.40	24.40	125	0.009	0	0.000	19	0.025	0.0013	10:00:00	0:00:00
MON, AUG 24, 87	0.61	97.29	124,754	0.14	0.52	27.77	153	0.009	0	0.000	20	0.026	0.0012	10:00:00	0:00:00

D A I L Y C I C S A C T I V I T Y R E P O R T

SYSTEM SUMMARY AND AVERAGES (BY HOUR)

CICS SYSTEM ID=KCP1 DATE=MONDAY, AUGUST 24, 1987

HOUR	OBJ MISS	RESP AVG	PCT <3S	TRANS	TRANS /SEC	PAGEIN /SEC	PAGEIN /CPU	CPU /DISP	CPU /USER	CPU %	MAX %	VSAM /SEC	VSAM WAIT	DBMS /SEC	DBMS WAIT	JOUR /SEC	JOUR WAIT	CPU /IO	DOWN TIME
0		0.12	100	207	0.08	0.00	0.00	0.00	0.03	0	0	1	0.006	0	0.000	0	0.016	0.0020	0:00:00
1		0.18	100	217	0.06	0.00	0.00	0.00	0.03	0	0	1	0.008	0	0.000	0	0.015	0.0013	0:00:00
2		0.18	100	324	0.09	0.00	0.00	0.00	0.02	0	0	2	0.008	0	0.000	0	0.013	0.0013	0:00:00
3		0.06	100	103	0.03	0.00	0.00	0.00	0.05	0	0	0	0.003	0	0.000	0	0.013	0.0011	0:00:00
4		0.22	99	198	0.06	0.00	0.00	0.00	0.03	0	0	1	0.007	0	0.000	0	0.014	0.0013	0:00:00
5		0.14	100	439	0.12	0.00	0.00	0.00	0.02	0	0	1	0.006	0	0.000	0	0.015	0.0014	0:00:00
6		0.19	100	2,096	0.58	0.00	0.00	0.01	0.02	1	2	6	0.009	0	0.000	0	0.015	0.0016	0:00:00
7		0.23	100	11,100	3.08	0.00	0.00	0.08	0.02	6	10	43	0.007	11	0.000	3	0.018	0.0014	0:00:00
8		0.29	100	25,366	7.05	0.45	1.85	0.48	0.04	24	28	116	0.008	9	0.000	9	0.023	0.0015	0:00:00
9		0.34	99	20,270	5.63	1.03	4.05	0.56	0.06	25	29	83	0.010	6	0.000	9	0.022	0.0018	0:00:00
10		0.35	99	23,690	6.58	1.06	4.68	0.47	0.05	23	28	90	0.011	7	0.000	8	0.023	0.0018	0:00:00
11		0.33	99	17,142	4.76	1.82	8.58	0.46	0.06	21	27	70	0.010	12	0.000	7	0.022	0.0016	0:00:00
12		0.24	100	13,156	3.65	0.05	0.33	0.19	0.04	14	18	49	0.008	13	0.000	5	0.018	0.0015	0:00:00
13	+	3.70	96	18,200	5.06	1.44	7.93	0.35	0.05	18	21	61	0.012	4	0.000	6	0.023	0.0019	0:00:00
14		0.30	99	20,236	5.62	1.01	5.29	0.35	0.04	19	19	59	0.014	6	0.000	9	0.020	0.0019	0:00:00
15		0.35	99	21,711	6.03	1.35	5.69	0.53	0.06	24	29	70	0.013	21	0.000	8	0.021	0.0018	0:00:00
16		0.22	100	7,774	2.16	0.49	5.02	0.15	0.06	10	19	18	0.012	3	0.000	2	0.018	0.0018	0:00:00
17		0.15	100	2,524	0.70	0.01	0.42	0.03	0.04	2	3	10	0.007	0	0.000	1	0.018	0.0015	0:00:00
18		0.20	99	1,419	0.39	0.00	0.01	0.02	0.05	2	2	9	0.007	0	0.000	0	0.018	0.0014	0:00:00
19		0.12	100	1,653	0.46	0.00	0.06	0.01	0.03	1	3	5	0.005	0	0.000	1	0.018	0.0014	0:00:00
20		0.17	100	483	0.13	0.00	0.14	0.01	0.04	1	1	3	0.004	0	0.000	0	0.017	0.0011	0:00:00
21		0.29	100	2,517	0.70	0.00	0.12	0.01	0.01	1	1	3	0.003	1	0.000	0	0.016	0.0013	0:00:00
22		0.18	100	425	0.12	0.00	0.00	0.00	0.03	0	1	2	0.008	0	0.000	0	0.016	0.0013	0:00:00
23		0.24	100	499	0.14	0.00	0.00	0.01	0.04	1	1	2	0.009	0	0.000	0	0.017	0.0013	0:00:00
TOTAL				191,769															0:00:00

DAILY CICS ACTIVITY REPORT

TRANSACTION SUMMARY

("X" TRANSACTIONS EXCLUDED FROM RESPONSE TIME SUMMARY)

CICS SYSTEM ID=KCP1 DATE=MONDAY, AUGUST 24, 1987

TRAN		COUNT	RESP			CPU			VSAM			DBMS			JOURNAL			TERMINAL		
			AVG	MAX	<3S	AVG	MAX	/IO	TOTAL	WAIT	/TRAN	TOTAL	WAIT	/TRAN	TOTAL	WAIT	/TRAN	WAIT	SUSP	I/O
AA00		966	0.11	2.69	100	.015	.057	.015	1004	.012	1	0	.000	0	4	.021	0	0.00	0.00	3251
AA01		36	0.07	0.25	100	.015	.025	.003	174	.006	5	0	.000	0	0	.000	0	0.00	0.00	4933
AA02		131	0.32	3.18	99	.032	.177	.002	1901	.007	15	0	.000	0	218	.024	2	0.00	0.00	4945
AA03		1,715	0.27	6.57	100	.028	.097	.003	10098	.021	6	0	.000	0	4949	.020	3	0.00	0.00	4927
AA04		1,138	0.16	3.77	100	.022	.083	.002	13604	.006	12	0	.000	0	0	.000	0	0.00	0.00	4940
AA05		75	0.17	1.68	100	.024	.047	.002	1003	.005	13	0	.000	0	0	.000	0	0.00	0.00	4953
AA06		4	0.18	0.43	100	.024	.041	.024	4	.023	1	0	.000	0	0	.000	0	0.00	0.00	4897
AA10		289	5.77	238.1	94	.069	.278	.002	6639	.017	23	166	.000	1	2076	.022	7	0.00	4.01	5020
AA12		16	0.92	1.75	100	.092	.162	.002	591	.012	37	0	.000	0	140	.018	9	0.00	0.00	5156
AA14		690	0.61	13.52	98	.056	.326	.002	16020	.015	23	0	.000	0	3421	.023	5	0.00	0.00	4943
AA16		514	0.63	7.45	99	.054	.270	.002	11316	.012	22	0	.000	0	4527	.025	9	0.00	0.00	4923
AA18		19	0.22	0.48	100	.029	.053	.004	104	.017	5	0	.000	0	45	.018	2	0.00	0.00	4914
AA22		29	0.78	3.36	97	.063	.088	.002	498	.019	17	0	.000	0	233	.023	8	0.00	0.00	5014
AA24		1,065	0.93	8.50	98	.076	.303	.002	34053	.015	32	0	.000	0	11730	.021	11	0.00	0.00	4967
AA31		613	0.09	1.40	100	.023	.053	.001	9873	.001	16	0	.000	0	0	.000	0	0.00	0.00	4917
AA32		887	0.13	3.16	100	.018	.097	.003	5356	.008	6	0	.000	0	0	.000	0	0.00	0.00	4940
AA33		386	0.13	5.55	100	.016	.072	.006	1005	.014	3	0	.000	0	0	.000	0	0.00	0.00	4955
AA40		843	0.17	2.10	100	.017	.094	.005	2893	.031	3	0	.000	0	0	.000	0	0.00	0.00	4917
AA41		8	0.08	0.26	100	.017	.036	.011	12	.015	2	0	.000	0	0	.000	0	0.00	0.00	4939
AA42		57	0.11	0.85	100	.017	.042	.006	163	.011	3	0	.000	0	0	.000	0	0.00	0.00	4915
AA43		45	0.17	1.34	100	.016	.046	.005	140	.008	3	0	.000	0	0	.000	0	0.00	0.00	4941
AA50		445	0.23	3.57	100	.021	.063	.002	3346	.010	8	0	.000	0	703	.023	2	0.00	0.01	985
AA60		1	1.05	1.05	100	.018	.018	.	0	.000	0	0	.000	0	0	.000	0	0.00	0.00	438
AA61		1	0.72	0.72	100	.031	.031	.005	3	.047	3	0	.000	0	3	.010	3	0.00	0.00	495
AA70		262	0.23	4.45	100	.019	.045	.002	3078	.008	12	0	.000	0	4	.010	0	0.00	0.00	522
AA71		281	0.13	3.49	100	.014	.043	.002	1908	.008	7	0	.000	0	0	.000	0	0.00	0.00	1751
AA72		19	0.10	0.23	100	.011	.019	.011	20	.028	1	0	.000	0	0	.000	0	0.00	0.00	532
AA73		32	0.50	5.18	97	.031	.182	.004	147	.061	5	0	.000	0	96	.022	3	0.00	0.03	685
ABRS		206	0.16	4.45	99	.012	.055	.005	84	.057	0	0	.000	0	385	.020	2	0.00	0.00	475
AB00		230	0.28	2.00	100	.020	.165	.004	995	.031	4	0	.000	0	308	.018	1	0.00	0.00	476
AB01		2,262	0.60	25.89	99	.043	1.48	.003	21177	.029	9	0	.000	0	11120	.021	5	0.00	0.00	545
AB02		8	0.33	0.58	100	.027	.036	.004	17	.023	2	0	.000	0	43	.017	5	0.00	0.00	506
AB03		49	0.60	6.54	96	.029	.049	.003	116	.034	2	0	.000	0	321	.027	7	0.00	0.00	501
AB05		26	1.32	5.94	81	.059	.178	.002	808	.033	31	0	.000	0	150	.019	6	0.00	0.00	597
AB06		3	0.42	0.68	100	.026	.028	.004	12	.041	4	0	.000	0	9	.011	3	0.00	0.00	491
AB07		4	0.25	0.65	100	.016	.023	.006	11	.016	3	0	.000	0	0	.000	0	0.00	0.00	479
AB08		36	0.12	0.82	100	.010	.023	.009	41	.043	1	0	.000	0	0	.000	0	0.00	0.00	518
AB09		154	0.12	1.49	100	.014	.043	.003	638	.011	4	0	.000	0	0	.000	0	0.00	0.00	490
AB11		175	0.73	63.96	98	.045	3.96	.001	3730	.016	21	0	.000	0	1869	.024	11	0.00	0.00	519
AB20		17,113	0.49	240.7	97	.048	3.63	.002	280128	.012	16	0	.000	0	73721	.019	4	0.00	5.02	6488
AB21		25	0.39	4.95	96	.015	.033	.008	22	.018	1	0	.000	0	24	.018	1	0.00	0.00	567
AB42		157	0.35	4.05	99	.017	.099	.009	316	.063	2	0	.000	0	0	.000	0	0.00	0.00	559
AB65		9	0.19	1.00	100	.015	.027	.003	51	.001	6	0	.000	0	0	.000	0	0.00	0.00	542
AB97		6	0.25	0.90	100	.012	.021	.004	2	.024	0	0	.000	0	18	.002	3	0.00	0.00	548
AB99		8	0.26	0.72	100	.014	.026	.007	16	.045	2	0	.000	0	0	.000	0	0.00	0.00	475
BEER	X	4	1043	2608	4	.025	.052	.003	28	.031	7	0	.000	0	8	.022	2	0.00	1041	0
BEIC		528	0.09	2.38	100	.013	.044	.002	3613	.005	7	0	.000	0	42	.022	0	0.00	0.00	1155

AN INTRODUCTION TO PROGRAM-TO-PROGRAM COMMUNICATION AND PEER-TO-PEER NETWORKING IN AN IBM SNA ENVIRONMENT

Thomas W. McDonald
MGT Technology Group
Natick, MA

ABSTRACT

Systems Network Architecture (SNA), IBM's communications architecture, is currently undergoing major enhancements which allow for the support of distributed transaction processing, program-to-program communication, and peer-to-peer networking. This paper looks at the developments which precipitated the new communications requirements, and the major architectures and products that facilitate these enhancements, including Logical Unit 6.2(Advanced Program-to-Program Communication), Physical Unit 2.1(Low-Entry-Network), APPN(Advanced Peer-to-Peer Networking), and related office systems applications.

1. INTRODUCTION

Historically, users of information systems have submitted specific transactions to be executed on centrally located mainframe computers. Today, as the nature of both the business and technological environments becomes more complex, users look toward the submission of business problems (rather than simple transactions) to dispersed information systems and resources. The challenge facing data processing and networking is to leverage all of the various resources or systems to provide integrated business solutions to the user.

This paper discusses the changes in the network services environment that are necessary for the information system to meet these user needs.

Since 1974 when IBM first announced SNA, the architecture has evolved to meet the needs of the user of traditional hierarchical SNA networks. In this traditional network, a user located at an unintelligent workstation communicates in a master/slave relationship with a centralized processor and information source. A typical SNA network has consisted of one or multiple mainframe processors, centralized databases, and a large number of dispersed terminal users (e.g. users stationed at 3270 CRT's or printers). These terminals are typically located at unintelligent workstations, (e.g. 3174 or 3274 cluster control nodes). (See figure A.)

The user of a terminal at an unintelligent, or **limited function work station** (**LFWS**), is heavily reliant on a central processor for application processing, message route determination, and management control functions such as component activation and deactivation. This hierarchical master/slave relationship between mainframe processors and LFWS's adequately serves the needs of users conducting inquiry/response work to centralized points of information.

However, the traditional 3270 user who wishes to step beyond the master/slave, inquiry/response profile and into the realm of distributed transaction processing and peer-to-peer communications, finds him or herself limited by the incompatibility of communication protocols between different products and the nonexistence of local processing and local network management.

2.0 FORCES DRIVING THE NEW ARCHITECTURE

The end-user's demand for the network to provide services beyond the traditional boundaries of the master/slave relationship has been primarily fueled by two forces: the replacement of the LFWS by intelligent workstations, and corporate and MIS management's perception of the network as a strategic tool. These two forces have come together to breed new access to information for the network end user.

2.1 The Intelligent Workstation

While the 3270 remains the mainstream terminal product populating SNA networks, recent changes in end user requirements outreach the abilities of the 3270, its datastream, and the architecture which supports it. The limitations of the 3270, coupled with breakthroughs in microprocessor technology have caused the replacement of many 3270's by stand-alone microprocessors such as personal computers(PC`s), and by multi-user departmental systems such as the IBM System/36, System/38, 9370, 5520, 8100, the DEC VAX and other minicomputers. This paper will generically refer to stand-alone microprocessors and multi-user departmental systems as **intelligent work stations** (**IWS**).

The IWS brought processing power and data storage and retrieval capabilities to the departmental and desktop level of operations. As the IWS began to populate SNA networks, users became aware of the potential for greater flexibility and for the desire for interconnection between systems. For example, a user of a personal computer may wish to have the independence of a stand-alone processor but at the same time have the ability to share files with other PC's and to access departmental systems, as well as maintain the mainframe access he or she experienced as a 3270 user. A departmental system such as a System/38 may wish to have access to other departmental systems for resource and database sharing, have access to a centralized mainframe database for periodic information updates, as well as serve as a departmental processor for its directly attached devices.

The SNA architecture provides services to network users in the areas of applications support, logical services support and physical services support. As the IWS begins to take a predominant place in SNA networks, the profile of network usage changes. This support of applications and physical and logical services had to be expanded to support the users at these intelligent stations.

2.2 CORPORATE MANAGEMENT'S MOTIVATION

The second force that drove the need for major enhancements to the network architecture was management's growing perception of the network as a strategic tool within the corporation. Corporate management recognized a competitive edge in providing "any-to-any" functionality by incorporating the separate and specialized networks into a single homogeneous corporate network, while at the same time retaining departmental autonomy. This motivation to produce an integrated, homogeneous environment was propelled further with the perception of the "extended corporate environment", i.e. the idea that for maximum competitive edge, the corporate network should reach out to include the company's suppliers, sellers, and customers.

As the view of the network as an integrator which ties together disperse groups throughout the company and extended enterprise grows, the ability to provide accessibility to corporate wide resources for any user across a homogeneous network becomes a central deliverable of network services. This means that a user at a terminal or an IWS must be presented with a single window to data, corporate-wide.

3.0 A NEW NETWORKING ENVIRONMENT

The local availability of processing power and data retrieval capability that the IWS brought to SNA, along with the new requirements for end user flexibility which resulted from it, came together with management's perception of the network as a strategic resource to breed new distributed applications and service needs. These include: decision support systems and office system applications, such as message delivery, document exchange, and distributed library functions; the need for centralized network management; and the ability to take advantage of the cost savings inherent in the integration of separate networks, especially voice and data integration.

These new applications and service needs spelled trouble for an architecture that was designed to support communications in a hierarchical fashion. In a network which includes IWS's, intelligence is no longer located at a central serving point but is distributed among IWS's.

An IWS looks much more like an equal to a mainframe processor than a slave. It contains a processor, an operating system and data storage capability. A network architecture that provides for a master/slave relationship only, will either not allow the IWS to exercise all of it's capabilities, or force the end user to write their own communication support services. In the integrated network, both hierarchical as well as peer to peer communication must be supported. (See figure B.) As a peer, the IWS would be able to independently initiate and terminate communication with another user, as well as control processes such as error recovery.

3.1 DISTRIBUTED TRANSACTION PROCESSING

To take advantage of the power of the IWS, processing power and information banks are distributed away from a centrally located processor to IWS's that are local to the end user and which may be targeted for specific departmental functions. This use of IWS's and the related ability to distribute intelligence between nodes leads to the ability to execute distributed transaction processes. The new application requirements stated above execute by **distributed transaction processing.** It is the need to support distributed transaction processing that is the driving force behind the changes in SNA's logical and physical services and applications support.

Distributed transaction processing involves two or more transaction programs, cooperating via program intercommunication to share each other's resources such as processor cycles and data bases. Distributed transaction processing takes place when a business transaction, generated on an IWS or a mainframe computer, can be broken into components and separate components can be executed on separate targeted machines. A machine may be targeted because of processor function, data handling or retrieval abilities, or physical location. These independent executions are brought back together at an IWS or a mainframe and a single presentation image is presented to the end user.

3.1.1 AN EXAMPLE OF DISTRIBUTED TRANSACTION PROCESSING

Figure C shows a possible network configuration for an insurance company. Remote field personnel use a personal computer to communicate with a regional departmental System/38 for regional customer information, a System/36 for price quote information, and a System/370 for policy forms and issuance.

In this example, the field worker needs to respond to a request from a potential client for an automobile policy. An application in the PC responds to the field worker's query by initiating multiple transactions.

The PC initiates a transaction with the System/38 requesting customer history information; a second transaction is

A HIERARCHICAL SNA NETWORK

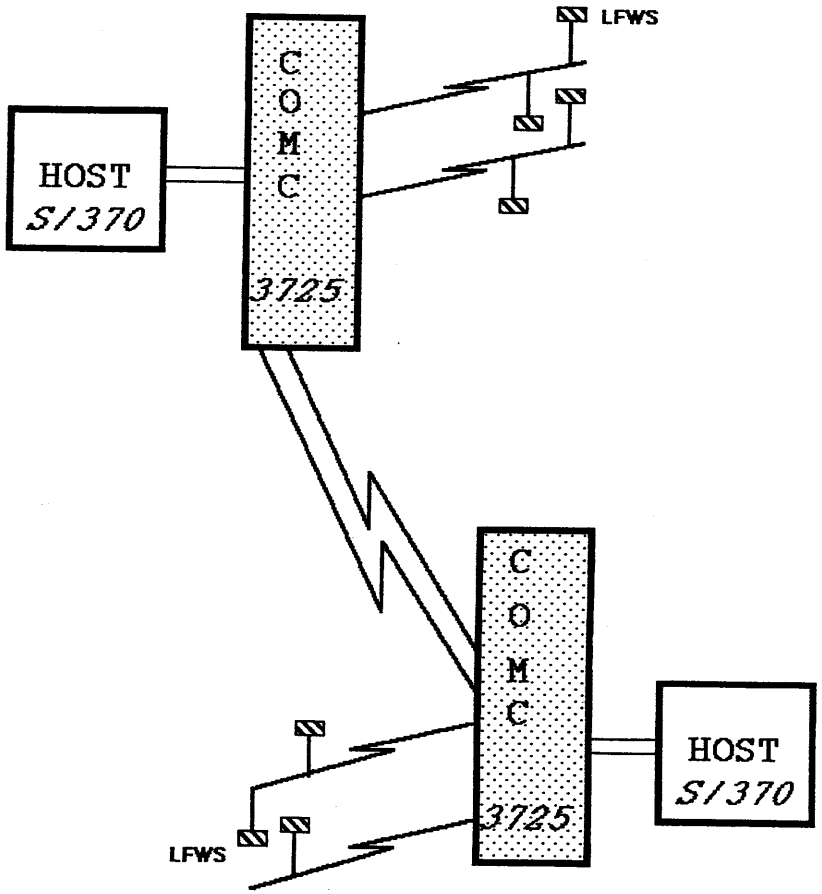
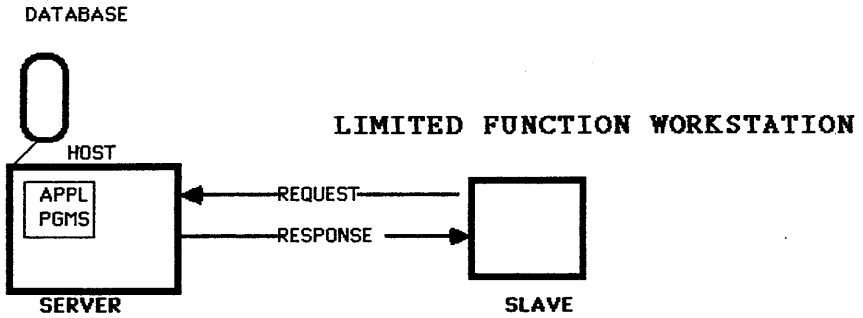


FIGURE A

MASTER/SLAVE



PEER-TO-PEER

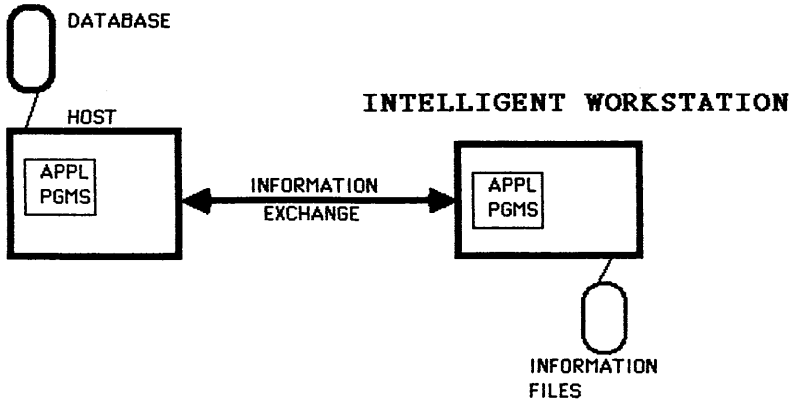


Figure B

DISTRIBUTED TRANSACTION PROCESSING

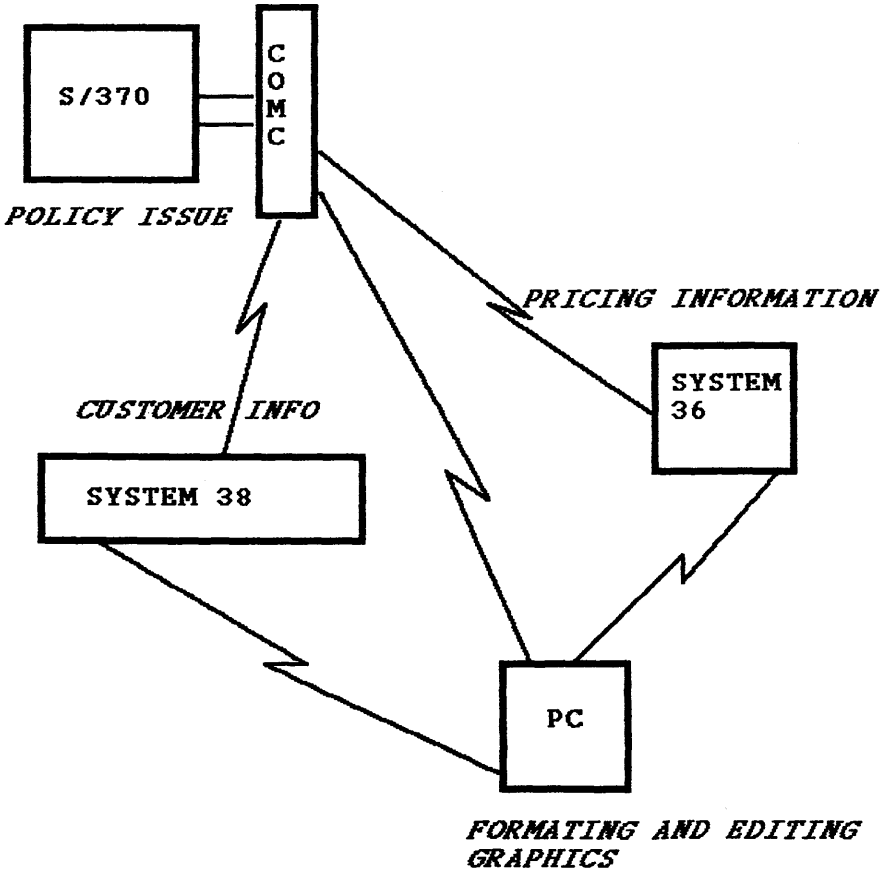


Figure C

CONVERSATIONS AND SESSIONS

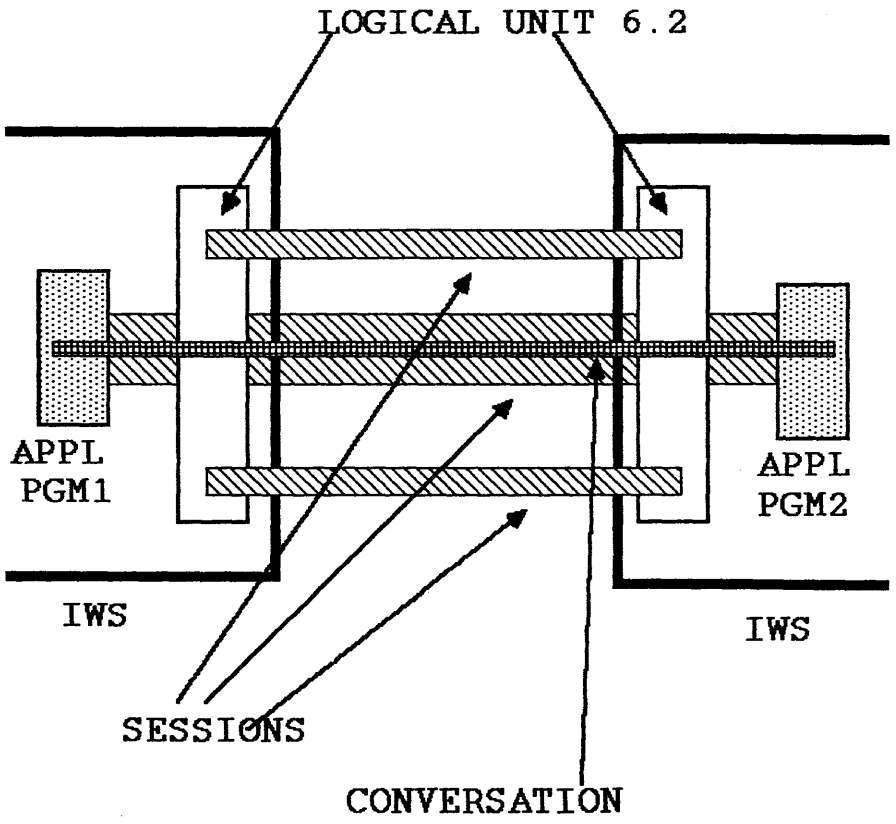


Figure D

initiated to the System/36 for price quotes. (The System/36 may respond with a request for the customer information that the PC has received from the System/38 before it responds with the pricing information.)

Once the PC has received the pricing and customer information, it forwards it to the System/370 for policy issuance. The System/370 issues the policy and sends the information to a printer. The System/370 sends notification to the PC that the policy has been written and indicates the location of the printer.

In a distributed transaction processing environment the concept of an end-user has changed from the traditional sense. In a distributed transaction, the end user is often represented by an application transaction program rather than by a user at a terminal as it is in a hierarchical network. The interchange of information is between application transaction programs on distributed processors.

3.2 NEW SUPPORT NEEDS

The SNA communications needs that exist for a network which supports IWS's as distributed processors are dramatically different than those between mainframe application and LFWS. A communication architecture which provides physical, logical, and application services to meet these needs must provide for the following:

1) Support of peer-to-peer communications between distributed processors. This allows communication to take place within SNA between intelligent stations without the intercession of a mainframe computer.

2) Support of common communications protocols between intelligent stations. This allows users at mainframes, departmental processors, and PC's to readily communicate with one another.

3) Standard application program interfaces to network services. This allows applications developed across different processors and products to readily communicate with one another across a common network.

4.0 MEETING THE NEEDS OF THE NEW NETWORK ENVIRONMENT

These three major needs are answered in SNA by the definition of new logical unit and physical unit types. Brief definitions follow:

Logical Unit - A port through which an end user accesses the SNA network in order to communicate with another end user. Communications services are accessed by the end user via the logical unit.

The SNA logical unit provides the port between the transaction programs and the communication path which ties them together. It allows a transaction program to invoke remote programs and to exchange data between them.

Physical Unit - The SNA resource in each node responsible for managing and monitoring the subordinate resources attached to the node. Examples of nodes are a host processor, 3725 COMC, 3274 terminal controller.

4.1 LOGICAL UNIT 6.2

It is **logical unit 6.2** that provides the common communications protocols which facilitate the coordination of processing between distributed transaction applications and the transfer of data between users of different systems. Logical unit 6.2 was designed specifically to support communication between distributed applications programs. It defines communication protocol for moving data between users and provides services that coordinate data exchange and flow between distributed applications.

IBM markets this facility under the name of **Advanced Program-To-Program Communication (APPC)**. Logical unit 6.2 and APPC are used interchangeably in current literature.

4.1.1 STANDARD INTERFACE BETWEEN THE APPLICATION AND THE LOGICAL UNIT

One of the attributes which makes logical unit 6.2 unique against previously defined logical unit types is its support of an architectured verb interface. A transaction program uses logical unit 6.2 to communicate with other transaction programs by issuing transaction program verbs.

This verb interface provides a common way for applications to request services of logical unit 6.2. It allows a transaction application written on one distributed system to use logical unit 6.2 protocols and to operate compatibly with an application on another distributed system or on a mainframe which also uses logical unit 6.2. For example, an application which wishes to establish communication with a second remote application would issue an **Allocate** verb to logical unit 6.2. The logical unit would then establish communication with the second application by the initiation of a conversation between the two users.

The capabilities within logical unit 6.2 can be broken into subsets of a base set of functions and optional sets. All products which support logical unit 6.2 must support the base set of protocols. This insures a common communications protocol between logical unit 6.2 stations. (A list of the basic set of verbs can be found in Table one. Each verb maintains a parameter set).

Additional functions such as advanced recovery procedures, specialized error detection, and security routines are implemented on an individual product basis

by the issuing of optional verbs. The use of base or option sets is communicated by the distributed transaction applications who wish to transfer information by an information exchange that takes place at SNA session initiation. This insures compatibility of protocols between the conversing applications.

4.1.2 ESTABLISHING COMMUNICATION VIA LU6.2

Another important aspect of logical unit 6.2 is its ability to support:

1) Multiple sessions between logical units.
2) Multiple conversations over a single SNA session.

A **session** can be defined as an exclusive logical connection between users in an SNA network. In order to communicate, users must be "in session". In hierarchical SNA networks, end users can support a maximum of one session at any one time. When a user wishes to break communication with one user and start communication with another, the session with the first user must be brought to an end, and a new session initiated with the second user.

As end users add function, this single-session limit becomes restrictive to smooth operations. As the terminal user is replaced by an IWS, the restrictions seriously limit the practical application of the IWS.

To illustrate this problem consider an example in which an end user is an application on a System/36 supporting multiple terminals. Multiple users may be requesting that the System/36 application be transferring data between a local application and other remote applications. If only one communication could be occurring at a time, and if communication had to be broken and reestablished for each transfer of information for a different user, a performance bottleneck would likely occur.

To prevent such a bottleneck from occurring, logical unit 6.2 allows multiple sessions to exist concurrently between users. Users use the session as a resource. They establish communications across that session by establishing a **conversation.** Conversations may be interleaved over a single session without breaking and reestablishing communication.

A conversation can be thought of as a session within a session. It gives the end users exclusive use of the session for the duration of the conversation. It is a conversation that the logical unit initiates in response to the Allocate verb from the application. When a deallocation takes place, the conversation is terminated and another conversation may be established by the same users or other users on this session. This allows communication to be established and broken without session binding and unbinding overhead.

The exclusive relationship which tied a particular session to specific users no longer exists. The logical unit 6.2 session is now a shared resource. (Figure D illustrates the uses of sessions and conversations between users of logical unit 6.2.)

Logical Unit 6.2 is transparent to the protocol of the communications link. It is therefore compatible with IBM's Synchronous Data Link Control(SDLC), X.25, and local area network(LAN) and private branch exchange(PBX) protocols.

4.1.3 SUMMARY OF LU 6.2 CAPABILITIES

Logical unit 6.2 facilitates communication between users of the integrated network by:

1) providing a standard verb interface between transaction application programs and the logical unit services. This provides access to communication services that establish and coordinate data exchange between distributed applications.

2) providing common sets of communication attributes between distributed users. This allows communication to take place between users in disimiliar environments.

3) facilitating interleaved conversations and multi-sessioning.

4.2 PHYSICAL UNIT 2.1

Physical unit 2.1 allows peer-to-peer communications to take place. It does this by :

1) providing connectivity enhancements

2) facilitating local network management and control

4.2.1 CONNECTIVITY ENHANCEMENTS THROUGH PU2.1

Physical unit 2.1 allows remote IWS's to directly link-attach to one another and provides multiple link support for the IWS.

To carry out peer-to-peer communications it is necessary to have the ability to directly connect IWS's to one another and to have the ability to connect to more than one station at a time.

The most common SNA control units, the IBM 3174 and 3270 family of cluster controller nodes, are SNA physical unit 2.0's. The PU 2.0 supports only one link attachment and this attachment must be to a more intelligent device such as an IBM 3725 communications processor, or an SNA host. The physical unit 2.1 incorporates multi-link support into the workstation, as well as provides for the ability to connect with another physical unit 2.1, or with a communications processor or SNA host.

4.2.2 NETWORK MANAGEMENT/CONTROL ENHANCEMENTS WITH PU2.1

Physical unit 2.1 provides network management and control capabilities local to the IWS.

Traditionally, the SNA host has been the center of the network. It is from here that the hierarchical network is defined, operated, and managed. The component in the SNA host that handles these tasks is the **System Service Control Point (SSCP)**, which is resident in the host access method.

A subset of the SSCP functions has been moved into the physical unit 2.1 as the **Single Node Control Point (SNCP)**. (In some manuals this is referred to as the Network Node Control Point.)

The SNCP provides the distributed node with the ability to provide network management control such as activation and deactivation of network resources and the ability to resolve network addresses and conduct session establishment in a peer to peer context. This frees the distributed processor from dependence on the mainframe and the SSCP for communication management functions.

The implementation of physical unit 2.1 with logical unit 6.2 (also called **Low-Entry-Networking(LEN)**), puts in place a mechanism which allows IWS's to communicate within a full-function SNA network, in a peer-to-peer and application-to-application fashion without a mainframe processor.

An IWS using logical unit 6.2 has the option of communicating with a mainframe or with a peer workstation. To communicate with a mainframe, a logical unit 6.2 workstation does not need to implement physical unit 2.1. Currently, when a physical unit 2.1 communicates with a mainframe, it does so under control of the SSCP, and in a traditional master/slave manner (see figures E and F). The logical unit 6.2 capabilities in the host may reside in a VTAM application subsystem (currently CICS is the only subsystem offering this support), or as the following section discusses, this support will soon reside in VTAM.

4.2.3 VTAM SUPPORT FOR LU6.2

IBM recently announced logical unit 6.2 support within ACF/VTAM. VTAM is the communications access method resident in a mainframe computer. The SSCP resides in VTAM. This support means that all VTAM based application subsystems (e.g. IMS, CICS, TSO) will have logical unit 6.2 support through VTAM. Previously, only CICS maintained support. This support is scheduled to be implemented in the third quarter of 1988.

One important benefit of this support is that users at an IWS may now look upon their counterparts in a System/370 as peers. The IWS may behave as a primary or secondary logical unit in its communication with the S/370 and may maintain multiple sessions. This breaks the master/slave relationship between the S/370 and its users. The user may now view the S/370 as just another IWS. With this implementaion, the communication depicted in Figure B will become a reality.

4.2.4 SUPPORT FOR 3270 APPLICATIONS

As logical unit 6.2 applications are developed, 3270 applications will continue to serve as the dominant sources of information for network users for the near future. Logical unit 6.2 software does not block the users at the IWS from communicating with the 3270 applications in the mainframe. Without such dual capability, the user would be forced to choose between being able to access the "old" 3270 applications, or those of the new logical unit 6.2.

4.3 APPN

Logical unit 6.2 and physical unit 2.1 make no accommodations for communication between nodes not directly adjacent to one another. For users in a limited geographic region, one solution to allowing the integration of multiple nodes is the implementation of a local area network. For more dispersed networks, this function is provided by **Advanced Peer-to-Peer Networking (APPN)**.

In traditional SNA networks, network routing and topology changes are implemented in a rigid fashion. Hierarchical SNA networks route data along pre-fixed routes. Intermediate nodes do not route based upon a real-time best path available scheme. In a traditional SNA network, changes to the topology of the network (i.e. adding a new node, or reconfiguring existing nodes), can only take place when the entire network is in an inactive state.

An analogy is often made comparing this restriction to the phone company requiring every user of the phone service to hang-up each time the phone company added a new phone to the system.

In an integrated environment, networks may consist of users who are in non-related departments or companies. These units may be isolated from one another both geographically and organizationally, and they may not maintain high levels of technical support locally. In this environment it is unreasonable to expect that changes to a single user's environment must be considered by all users of an integrated network.

Advanced Peer-to-Peer Networking (APPN) brings dynamic routing and dynamic topology updating capabilities to SNA logical unit 6.2 networks. APPN maintains directories of users and locations and topology databases which allow for dynamic configuration of the network and dynamic routing of data to its destination by the most expeditious route. With APPN new

APPC WITH PHYSICAL UNIT 2.0

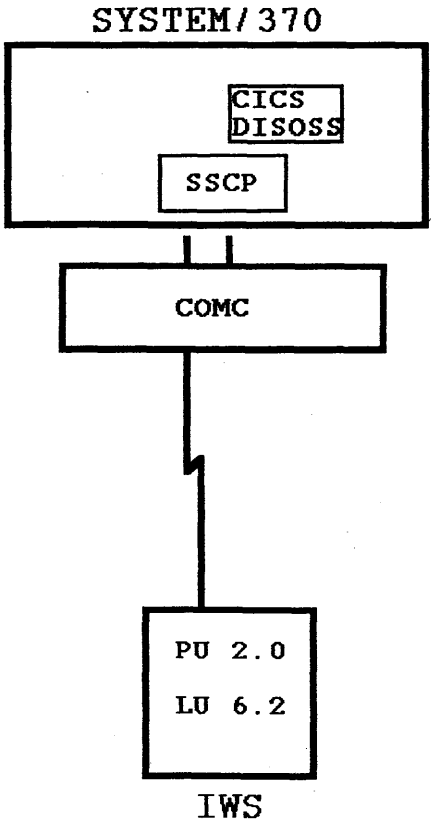


Figure E

APPC WITH PHYSICAL UNIT 2.1 PEER-TO-PEER NETWORKING

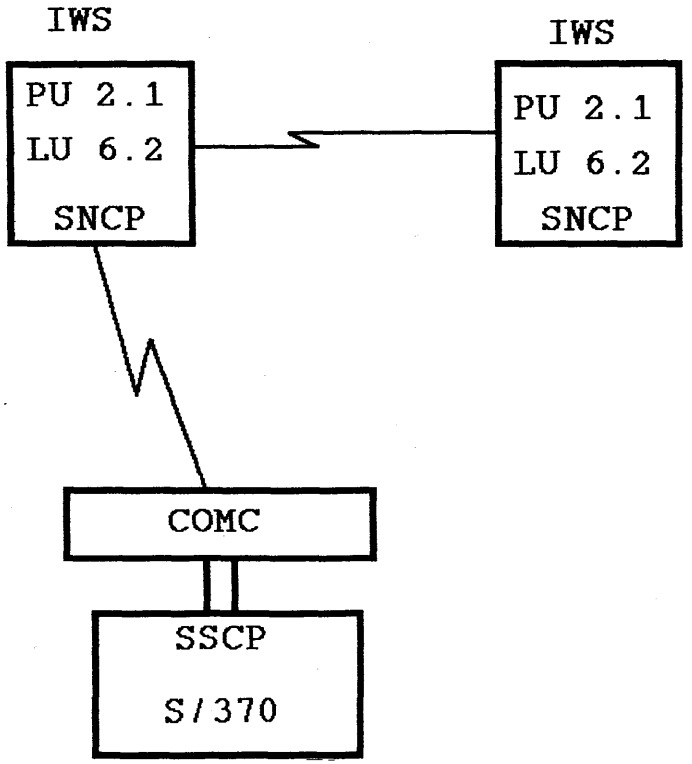


Figure F

users can be brought on-line transparently to the functioning network.

APPN is not an architecture. It is a feature of the System/36. It is an extension of physical unit 2.1. Only System/36's with physical unit 2.1 capabilities support APPN. See figure G.

System/36's which house APPN are termed "network nodes". They may support multiple "end nodes" or "peripheral nodes." (See figure H). Each network node maintains a table of its peripheral node entries. This information is not distributed throughout the network to other network nodes.

4.3.1 EXAMPLE OF DYNAMIC ROUTING WITH APPN

When a peripheral node requests its owning network node to communicate with a second peripheral node on a second network node, an undirected search takes place (see figure H). In this example peripheral node Jan T. is trying to establish communication with peripheral node Chris F. After Chris F. is located, an entry is made of its location and that of network node 3 in the cache of network node 6. Network node 6 sends the data to network node 3 via network node 5 (network node 5 has been resolved by APPN in network node 6 as the next node on the best route available to network node 3). Intermediate network nodes dynamically route data across the most expeditious route until the information arrives at the desired destination network node.

An alternative to an undirected search is to dedicate a network node as a directory node. All searches are then directed to the directory node. Cache entries are made in both the directory node and the node requesting the connection.

4.3.2 TOPOLOGY DATABASE AND DYNAMIC ROUTING WITH APPN

An example of how APPN allows topological changes to take place to the network without interrupting network operation follows. (See figure I).

Each network node maintains a topological database of the network. When a network node is added, the new node sends out a topological database unit (TDU) to all adjacent network nodes. The TDU contains information on the new nodes configuration attributes. The adjacent nodes reciprocate by returning TDU's of the existing network nodes. This TDU exchange ensures that all network nodes in an APPN network maintain a complete and current topology database.

APPN eliminates the need to pre-define routes between all users in a network. Each network node uses its knowledge of the network's topology to ensure that the most expeditious route will be taken by the data it is routing.

One consequence of maintaining this knowledge is a functional limit to the number of nodes that can realistically be supported in an APPN network. The limit is considered to be about one hundred nodes. This limitation indicates that APPN will not cause hierarchical networks to become extinct. They will coexist with them, operating as satellites, alternating between independent and integrated operations.

4.4 LU6.2, PU2.1, AND APPN WORKING TOGETHER

Logical Unit 6.2, Physical Unit 2.1, and APPN allow for the transparent interchange of corporate information irrespective of the location of the end users and of their hardware or software environments.

These new features allow:
1) The accommodation of distributed application processing.
2) Flexible and robust use of the IWS.
3) The corporate network to be viewed as a uniform resource capable of being used as a strategic tool.

5.0 APPLICATION SUPPORT

The majority of software available on logical unit 6.2 networks today consists of office systems software. These software tools work together to provide the ability to easily create, process, store , and distribute data, text, and video files. This paper will discuss : **Document Content Architecture(DCA)**, **Document Interchange Architecture(DIA)**, and **SNA Distribution Services(SNADS)**.

5.1 DOCUMENT CONTENT ARCHITECTURE

DCA defines a uniform formatting of documents that are to be interchanged in an office systems environment. DCA delineates the guidelines for formatting functions such as paginating, highlighting, centering,etc, for documents to be exchanged between office systems.

5.2 DOCUMENT INTERCHANGE ARCHITECTURE

DIA is a set of conventions that allows the distribution of office systems functions across diverse systems. It is an architected application that allows the exchange of files between office systems in a network. It carries out local document distribution, storage, retrieval, and transfer. It defines the data stream constructs for the exchange of information between two DIA users. Transaction programs are the "users" of DIA.

It is DIA which allows different IWS's such as the System/36, System/38, Series/1, 5520, Displaywriter, and DISOSS/370 to exchange office documents among one another when operating as office systems nodes.

DIA transaction programs communicate by establishing conversations over logical unit 6.2, since DIA has no facilities for delivering data between DIA users. In order to deliver the data between remote DIA users, DIA employs SNADS.

ADVANCED PEER-TO-PEER NETWORKING

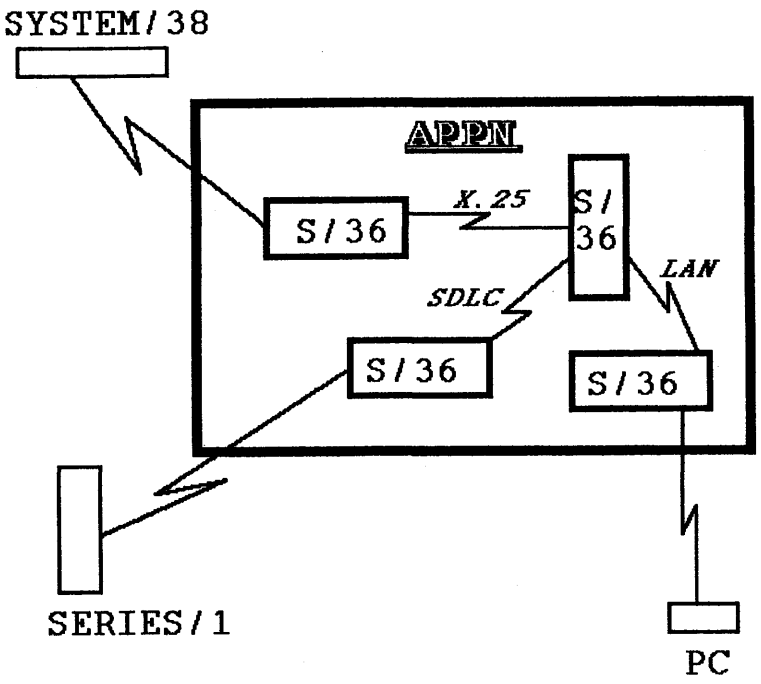


Figure G

APPN NETWORK UNDIRECTED SEARCH

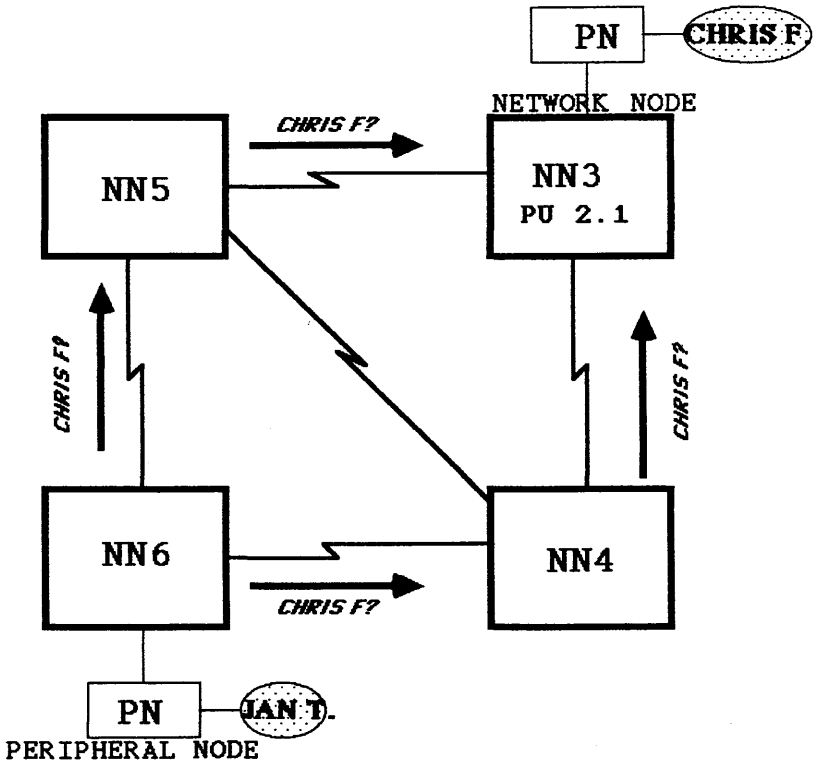


Figure H

APPN
DYNAMIC TOPOLOGY UPDATE

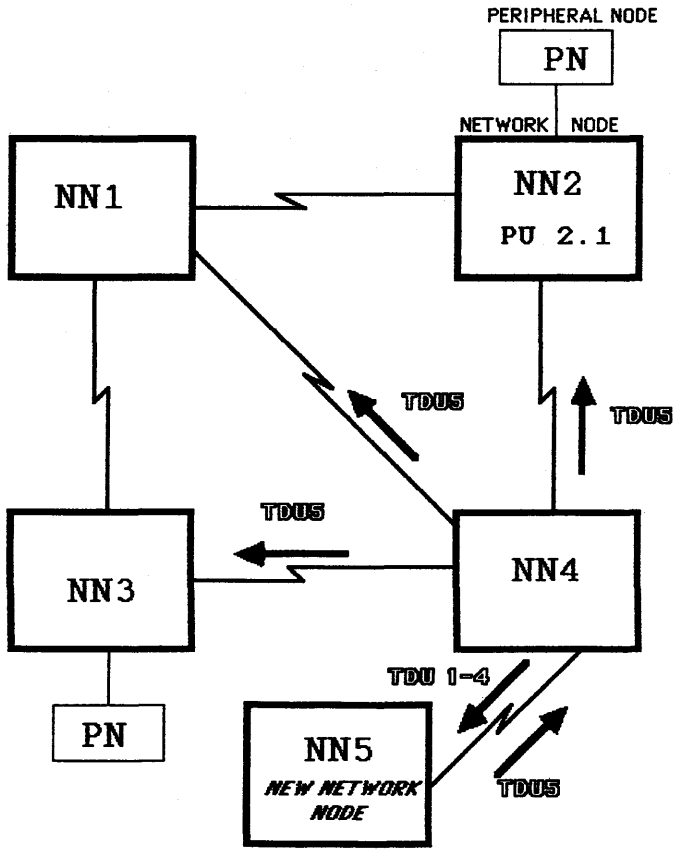


Figure I

SNADS NETWORKING

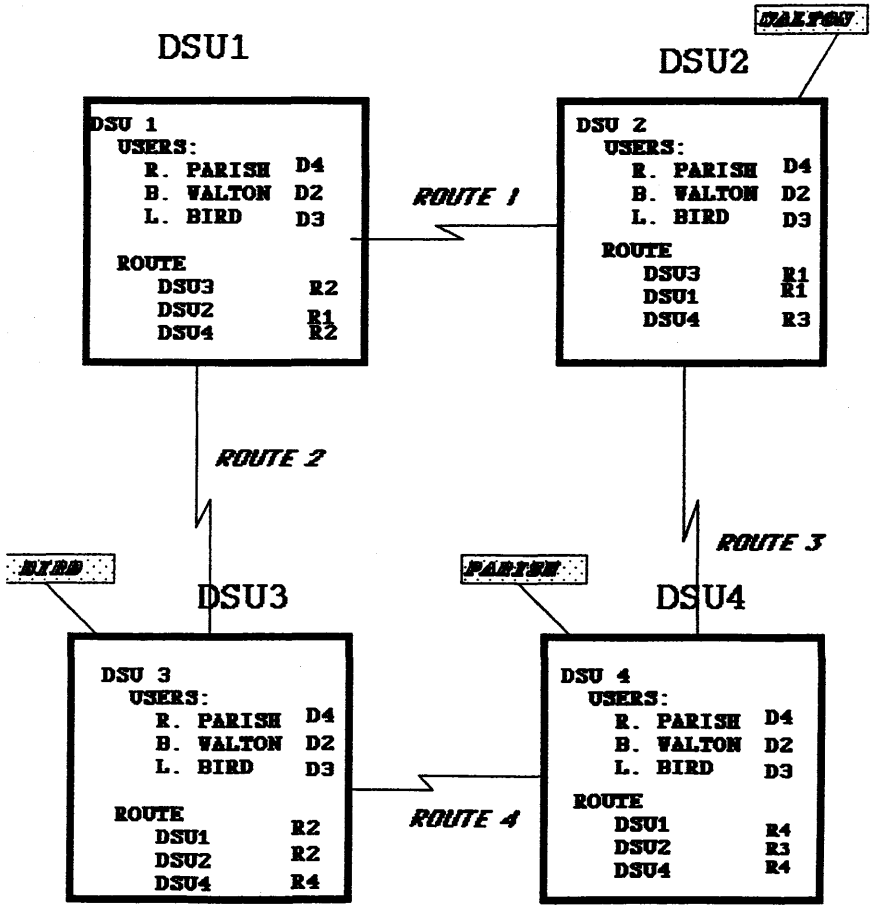


Figure J

DIA communicates with both logical unit 6.2 and SNADS by protocol boundary interfaces verbs. DIA documents follow DCA specifications.

5.3 SNA DISTRIBUTED SERVICES

SNADS is an architecture for the asynchronous distribution of information between network users. It provides a delivery system between IWS's serving as office system nodes. It accepts and delivers documents which are in DIA/DCA format.

Previous to SNADS, SNA supported the delivery of data in a synchronous fashion only. This means that the sender of the information, the receiver, and the components making up the path between them must be simultaneously active. (An example of "synchronous" communication is a telephone conversation.) For traditional hierarchical inquiry/response work this was adequate support. Distributed application programs, especially office systems programs, led to the need for support of an asynchronous system of delivery. Asynchronous delivery allows a sender or requester of information to do so without knowledge of the activity state of the receiver or the path between sender and receiver.

The SNADS function used with logical unit 6.2 and supporting software allows a user to interchange office systems documents to remote users without an awareness of the receiving users location, software environment, or the activity states of the receiving user or the components which make up the path between sending and receiving users. (An example of an "asynchronous" communication is the post office mail delivery system.)

SNADS consists of a set of transaction programs that run in a logical unit 6.2 environment. The SNADS "network" consists of interconnected distributed transaction programs that communicate synchronously via logical unit 6.2. This set of programs and individual synchronous communications form an asynchronous end to end delivery service.

A SNADS user is an addressable entity that can serve as the origin or destination of a file. They are termed "distribution users". Distribution users transfer data in the form of "distribution interchange units" (DIU's).

The requester of a file is termed the "originator user". The receiver of a file is termed the "destination user". Most products which implement SNADS have both originator and destination user ability.

Each distribution user has a unique name. The name is divided into two parts. The distribution group name (DGN) is unique to the SNADS network and the distribution element name (DEN) is unique to a group. These names map directly to the DIA assigned user names to facilitate transfer between SNADS and DIA.

The collection of transaction programs and data structures at a location that provides SNADS services are termed the Distribution Service Unit (DSU). Each DSU has a unique name. The routing group name (RGN) is unique to the SNADS network and the routing entry name (REN) is unique to a group.

Each DSU maintains a directory of entries of local and remote users. The DGN and DEN are resolved to their attached RGN and REN for routing. For large networks, it may be impractical for each DSU to maintain entries for all end users network wide. A practical subset of users could be maintained, with default routings listed for those that fall outside this subset.

5.3.1 SNADS EXAMPLE

An example of data transfer via a SNADS network follows. (See figure J.)

A user Bird at DSU3 wishes to send a memo to Walton in inventory control, and Parish in marketing services. DSU3 searches its directory and finds that the two targeted resources are in DSU2 and DSU4 respectively. It checks its route table to see what routes connect DSU3 with those remote DSU's.

A copy of the office document file is shipped across each of the routes to the destination DSU's. Any intermediate node goes through the same look up and routing procedure until the documents arrive at their destination DSU. The destination DSU hands the file to DIA for local delivery of the information or storage on local media if the targeted end user is currently inactive or unable to receive the data. An intermediate DSU may store the information in memory or on disk if the receiving or next DSU is unable to accept delivery. The intermediate DSU will attempt delivery when the receiving DSU becomes active.

In this example SNADS has conducted data delivery functions similar to those described for APPN. It should be noted that this ability is written specifically for users of a SNADS/office system environment. This service is carried out at the application level of the architecture. It is not a generic routing capability as is that provided by APPN, nor does it allow the dynamic reconfiguration that APPN does.

These office systems applications give us a good indication of how other data processing applications will be executed across the new architectures.

6.0 Summary

The networking environment and the services required to support the environment continue to change at a rapid pace. The addition of logical unit 6.2, physical unit 2.1, APPN, and associated supporting software to SNA completes the architectural framework for a communication environment

that will allow users of SNA networks transparent "any-to-any" access to information, irrespective of data location, communication protocol, or operating system environment.

This new architecture which allows for the coexistence and integration of users communicating in peer-to-peer and hierarchical fashions will allow the network to fulfill its role in providing solutions in a transparent manner to business problems posed by the information system user.

REFERENCES

1. GC30-3084-2, Transaction Programmer's Reference Guide for LU Type 6.2, IBM

2. SC30-3269-3, Format and Protocol Reference Manual: Architecture Logic for Lu Type 6.2, IBM

3. GG24-1693-0, IBM System/36 APPC Implementation Guide, IBM

4. GG24-1730-00, IBM System/36 in a SNADS Network, IBM

5. SC34-0608-1, APPC Programming Guide and Reference for Series/1, IBM

6. G320-0556-1, Document Content Architecture Interchange Compatibility, IBM

7. SC23-0763-1, Document Interchange Architecture: Transaction Programmer's Guide, IBM

8. GG24-1584-01, An Introduction to APPC, IBM

TABLE ONE
THE BASE SET OF LOGICAL UNIT 6.2 VERBS

ALLOCATE	Allocates a conversation with a remote logical unit
CONFIRM	Requests that the remote logical unit confirm receipt of data
CONFIRMED	Sends a confirmation of receipt to remote logical unit
DEALLOCATE	Releases the conversation between users at completion of transaction
GET_ATTRIBUTES	Retrieves SNA session parameters for a conversation establishment
RECEIVE_AND_WAIT	Waits for information to arrive on a conversation and then reads the data
REQUEST_TO_SEND	Notifies the remote application program that the local program wants to send data
SEND_DATA	Sends data to the remote application program
SEND_ERROR	Informs the remote logical unit 6.2 application program that an error has been detected

A Real-Life "How-to-do-it" Guide to Modelling

Michael Taylor
Bank of California
640 Battery Street Dept 126-05
San Francisco, CA 94111

Michael McElwee
Computer Measurement Services
7 Darnby Court
Orinda, CA 94563

Abstract

In this tutorial emphasis will be placed on the practical aspects of modelling from the user's perspective rather than that of the vendor or product developer. A straightforward "how-to" approach will be taken. Collection, summary and "shaping" of data will be emphasized so that the inputs to a model yield results that "make sense."

The authors feel that pragmatism and experience with the modelling process are the cornerstones of success. This tutorial is intended for those who have little or no actual modelling experience, who expect to become (more) involved in modelling, and who would like some of the mystery and esoterica associated with modelling dispelled.

We think that many of you are like us. We do not have advanced degrees in mathematics or statistics. We learned how to handle system measurements and models while on the job. Skills were acquired as needed in response to work assignments. Pragmatic application of those skills paid off. The ends justified the means; we utilized whatever tools and techniques were available to get the job done. In many respects this has been a process of trial and error.

For this reason we have adopted a pragmatic not a mathematical approach to the modelling process. We deal with queueing theory only to the extent that it is required to make informed judgments when selecting data and building the model. This tutorial attempts to give the flavor of a modelling study as it was created. The results are presented "warts and all" to give you a feel for the modelling process as a whole as it unfolds.

So, if you have yet to go through the process of modelling a system, or if you have a little experience and want to know if all the things that happened to you are "normal," then this presentation may be of some value to you. Our emphasis will be on creating and running models using software packages and on presenting results. Lots of formulas will not be presented. Little's Law will not be explained here.

By the way, copies of the foils are available. They were not included in the CMG '86 Proceedings because we felt that they were not sufficiently self-explanatory and that their content might be misconstrued. In addition, don't be upset if this presentation does not follow the write-up in the Proceedings word for word. Some of that is blather, anyway. All the good stuff is here, plus a great deal more.

We view modelling as a process. Since learning about a process is not the same as experiencing it, we have divided this tutorial into two parts, each with the same subdivisions. They are:

I. FIRST PASS: BACKGROUND AND BACKGROUND
II. SECOND PASS: THE REAL THING

In the first part we will talk about the modelling process. We will then make a second pass with a real model, warts and all.
Our modelling process has six subdivisions.

1. Tool selection.
2. Data collection.
3. Data formation.
4. Shaping the model.
5. Evaluating the model.
6. Interpreting the model.

The first item on our list affects the modelling process as a whole. Items 2 and 3 deal with creating the model, while items 4 and 5 have more to do with running and refining it. You will spend more time working with your data than you will with the model, especially the first time out. Later efforts will balance out differently. The final entry in our list considers how to present results to management and how management will evaluate the model. Sometimes there are surprises here - not all of them pleasant!

FIRST PASS: BACKGROUND & GROUNDWORK

Tool selection

Tool selection can affect the modelling process as a whole. There are several alternatives available. Mainframe-based products seem to be the most expensive, but offer the largest number of features. PC-based products are less expensive and have the advantage of offloading the host. They generally impose a data entry or transmission burden, however. You can program your model yourself using APL or SAS or some other language. Finally, you can do it by

hand with a good calculator and a pencil. which we believe to be normally distributed. We recommend mainframe or PC products designed to model your operating system environment(s). They provide much greater support for your efforts. The study presented in this tutorial was created from a commercially available PC-based modelling package.

Data collection

This is usually the toughest part of modelling, particularly the first time. There are a few rules and cautions in this area:

1. Know your system. You must be able to describe your system in terms of workloads and their servers, customers and queues.
2. Don't confuse detail with understanding. It is necessary to identify those phenomena in the system which are of interest. We need meaningful information not a mountain of facts.
3. Spend a lot of time choosing a good interval for the study. The purpose of the model is to examine the potential behavior of your system under particular conditions.
4. Decide how to distribute portions of shared servers to the workloads that you have some inkling about defining.

Data formation

This is where you formulate the workloads in your system. This is also where you must choose data from a particular time period as your starting point. The choice of interval is very important. The main problem that you will experience here is the nonexistence of a perfectly "representative" period during which time all of your workloads are in a steady state. You may find it advisable to choose to summarize several measurement periods into one for modelling purposes, or you may find it easier to find one small interval that works for most of your workloads. In the latter case, you know up front that you are going to have to "tweak" the model, a fact we all learn sooner or later anyway.

Remember that a model is merely a representation of reality - and an approximate one at that. So, it is important that workloads be defined so that they are reasonably homogeneous. A workload with a high degree of variation built int› its resource consumption patterns will not behave as predictably as one with less internal variability. In other words, take those "Jaws"-type transactions and make a separate workload out of them. We will have more to say about this in the second part of this tutorial.

Shaping the model

Here's where the real fun begins! A valid model should produce results that are within five percent of measured resource usage. Reported throughput and response times should be within ten percent at the outside of actual measurements. A well tuned model will reduce the throughput and response time margins to within five percent with concomitant reduction of resource usage variances. This, of course, is our mean opinion based on our experiences

If one or more of your workloads just doesn't look right, you should feel free to "fix" it so that it works. This may sound crude, but it really is not. The polite term for this activity of punching workloads where they hurt is **calibration**. A model that doesn't reproduce known results within reasonable bounds is not worth the keystrokes required to purge its output. As you will see, we forcibly calibrated a batch workload in our own model, and we are completely unrepentant.

Evaluating the model

This phase represents the bottom of a loop that may take you back to any of the preceding stages. If you have been careful, however, most of the time you won't go back past the shaping stage.

Your primary aim here is to be a beneficiary of the 80/20 Rule by finding the point at which the model is "good enough" as quickly as possible. That point is indicated when you have satisfactory answers to questions such as "Does the model answer the question(s) that made me build it?", and "Are the results within reasonable bounds?" When the yesses begin to pile up, quit fooling around and get to the last phase.

Interpreting the model

A model is a technical thing. A lot of expertise and insight go into its construction and ongoing use. You just can't take the output from a complex mechanism such as this and hand it over to a manager without a lot of explanation.

So, save yourself some trouble. Translate the results of your labors into a simple document that refers to the model as if it were a shrine worthy of veneration, but dismisses almost all of your work as a mere support effort dedicated to answering management's Great Question. Then provide information that can be used to back up a decision that involves spending a fair amount of money. For that is what management is concerned about most of the time.

SECOND PASS: THE REAL THING

Very little text is given here for this part of the tutorial, since the use of foils is critical and explanations of the foils' content is not practical.

Tool selection

Our choice of tool was dictated by cost and environmental considerations. There were pros and cons that affected our work.

Data collection

Data came from two main sources: RMF and Living Memory.

Data formation

39 servers, 6 workloads and a short interval were developed.

Shaping the model

Here we show what we changed and why in calibrating the model.

Evaluating the model

We looped a couple of times until it worked properly.

Interpreting the model

We succeeded. Management's views will be revealed.

BIBLIOGRAPHY

Allen, Arnold, Probability, Statistics and Queueing Theory, Academic Press, 1978.

Ferrari, Domenico, et. al., Measurement and Tuning of Computer Systems, Prentice Hall, 1983.

IBM, RMF Field Descriptions and Analysis Version 1. MVS/SP 1.3.0., IBM World Trade Systems Centers, Document Number GG24-1564-00, 1982.

Lazowska, Edward, et. al., Quantitative Systems Performance, Prentice Hall, 1984.

Merrill, H. W., Merrill's Expanded Guide to Computer Performance Evaluation Using the SAS System, SAS Institute, 1984.

ANALYTIC MODELING
FOR THE NON-ANALYTIC MODELER

Barbara Sannerud
Boole & Babbage

ABSTRACT

The objective of this tutorial is to provide a fundamental introduction to queueing flow problems. By relating to real world examples, this paper covers the basic laws of queueing theory including Little's law, response time law, forced flow law, and Markovian models. The development of queueing networks and multi-class models is also discussed.

Queueing theory, as a mathematical discipline, has its practical origins in the study of flow, such as traffic problems, network communications, assembly line problems, and so on. On a daily basis, we all encounter queueing theory when analyzing our time spent waiting in line at tollbooths, or when we get service at a gas station. Though not always evident, most large scale human endeavors are queueing flow problems. Queueing theory is also at the heart of capacity planning and is embodied in many of the basic rules for performance management.

Although capacity planning products use many of the mathematical models available, all are concerned with similar metrics and answer the same types of questions. The methodology used may involve simulation studies, benchmarking or analytic modeling. Analytic modeling is the technology used in many commercial capacity planning tools.

In spite of the fact that explanations of performance problems do not always neatly fit into existing analytic models, models are a basis on top of which refinements can be made. Unlike the science of programming, analytic modeling is not an exact science. Nevertheless, an understanding of queueing theory will help the analyst make more informed decisions and will help explain system performance.

Basic Queueing Theory

Regardless of where it is used, queueing theory addresses parameters such as arrival rates of customers, time a customer spends on line for service, and the service time the customer obtains. When relating queueing theory to practical problems such as transaction response time, think of transactions as customers arriving from either a finite or infinite source population. They may wait on a queue for service, or reneg, then they obtain service, and eventually they exit that server . The server may be an I/O device or the CPU or memory. They then move onto another queue or exit the system.

Example:

Take a stand alone environment with IMS transactions as the only customer and the CPU as the only server. If 100 transactions arrive over 10 seconds the transaction arrival rate is 10 per second; if 50 complete, the throughput is 5 per second. In the same 10 second interval, if the CPU is busy for 3 seconds then each transaction uses 3/50 or .06 seconds of CPU time. Given that the throughput is 5 transactions per second and that the average service requirements are .06 seconds, the utilization is 5x.06 or 30%. This relationship between throughput and service is the Utilization Law. Some of the basic formulae expressing these relationships are:

$\lambda = A/t$ A=arrivals t=Time λ =Arrival Rate
$X = C/t$ C=completions X=throughput
S=Service at the resource μ =utilization
$\mu = XS$

The Utilization of a resource is the product of the throughput of that resource and the service at that resource. This is known as the utilization law.

In a steady state environment, the product of the throughput of the system and the average time spent in that system by a request yields the number of requests present in the system at any given point in time. The same relationship holds true for customers in a queue awaiting service. This law known as **Little's law** is simply a modification of the utilization law and finds its way into many performance applications.

$N=XR$ or **$N_q=XR_q$** **N=number in the system X=throughput R=response time**

Where q denotes customers in the queue not receiving service.

TSO

There are times when Little's Law does not empirically hold true. In a TSO environment or in a lightly loaded batch environment, we might find discrepancies between actual and calculated results using these laws. What should response time be if there are 10 TSO users and TSO transactions are processed at a rate of 5/second? Since N=XR and 10=5R implies R=2, means response time should be only 2 seconds. This is internal time only, which does not consider think time. Little's Law is modified in this case for think time:

$N=X(R+Z)$ where **Z** is the average think time.This law is known as the **Response Time Law.**

While the Utilization and Little's law focus on a particular resource, the **Forced Flow** Law considers throughput and service across many different servers such as DASD and CPU. Essentially, it states that for all components in a system throughput rates remain proportional to one another. The visit count at a service resource equals the number of transactions completed at that particular resource compared with the total number of system transactions.

Example
If a typical IMS transaction does 10 reads and the DASD device services 20 I/O's per second attributable to IMS, then throughput is 20/10 or 2 transactions per second. If, at the same time, there are 8 buffers accessed per second from IMS, then each transaction requires 8/2 or 4 buffer accesses. The forced flow law can be used to determine throughputs at various servers. Throughput and residency at different service centers can be equated. Different servers in the same system tend to do proportional amounts of work in any given time interval.
The **Forced Flow Law** is expressed as:

$X_k=V_kX$ **X=throughput**
V=visits to a server
k=a particular resource

The throughput at resource 'k' is the product of the average number of visits a request makes to resource 'k', and the system throughput.

Example:

How can these laws be tied together to estimate response time? Take the following example given a system where there are:

•25 transactions in the system
•DASD utilization =35%
•service requirements at the device is 70ms. per request
•there are 10 visits to the DASD device per transaction:
What is response time?

Calculate response time:

•Utilization Law	$\mu=sx$	DASD throughput $x=\mu/s$ x=35%/70ms = 5 requests/second
•Forced Flow Law	$x_k=V_kx$	System throughput $x= x_k/V_k$ x=5 /10 =.5 interactions / second
•Little's Law	n=xr	Response time r=n/x r=25 / .5 or 50 sec.

Response time=50 seconds

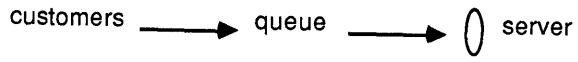
The Queueing Network Model

The previous discussions are useful for calculating response time on a small scale. But how do models predict response time in large computer systems? Essentially, one must identify the major bottlenecks or contention points from the universe of all possible service centers in the system. For example, I/O can be analyzed on a microscopic level, from the perspective of seek delays, set sector, wait for channel program, blocking factors, and so on. However, one must examine the major service centers or resources, and queues which have significant effect on your system.

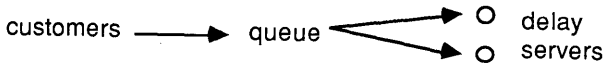
The modeling approach which examines the entire set of queues, service centers, customers and system resource contention is known as a *queueing network model.* The model becomes more complicated when you consider shared DASD since this involves contention from outside the system. A service center can be either a queueing center or a delay center.

The following diagrams illustrate three possible service configurations in a queueing network model.

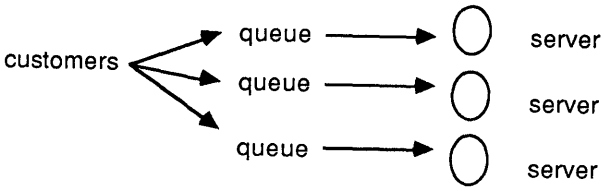
(1) A single server such as a UP processor



(2) A delay server such as think time for TSO



(3) Multiple servers such as tape or DASD devices



There are two major types of service centers in this model: a queueing center and a delay server. A delay server can be thought of as a constant or delay added to service. The time a transaction spends at a delay center is equal to its service requirements there. Customers each have their own dedicated server avoiding competition for service. There may be single service center (one cpu) or multiple service centers (DASD). Customers either receive service or queue for service; the two are mutually exclusive.Some of the parameters of queueing network models include :

(1) customers - described by arrival rate, population and delay service time
(2) service centers- described as queueing centers or delay centers
(3) service demands- resources consumed by customers at each center

•There are 3 types of customers: Open or transaction workloads like IMS, have variable size infinite source populations. Once a transaction receives service, it leaves the system. Batch or closed workloads have fixed populations and there is recirculation. Customers that leave the system are immediately replaced, i.e. job initiators service backlogs of requests. Terminal or TSO workloads are similar to both of the above, in that they have variable populations bounded by the number of active terminals but have think time and a variable number of transactions.

• Service centers were previously described as queue or delay centers.

•The service demands at each center are summed up for a customer. At each center, k, the customer demand is $D_k=V_kS_k$. The product of the number of visits to that server and the service requested at that service center, yield the service demand for that center.

Multiple Classes

The only major distinction between multiple classes and single class systems is the inclusion of a scheduling algorithm. A scheduling algorithm is a set of rules which determine which customers get serviced next and their duration of service. In a single class system, only 2 possibilities exist: queueing servers or delay servers. Multiple class systems exhibit many behaviors:

Scheduling Disciplines:

FCFS-first come first served. In FCFS, the first customer to arrive is served. The assumption is made that all classes have similar service requirements. There is no preemption of service.

LCFS-last come first served-preemptive. This is used to model high priority, frequently dispatched jobs. The arriving job preempts the current job until it has completed its service requirements. CPU's handle this algorithm by resuming preempted jobs. This algorithm is rarely used for DASD.

RR - round robin-time slicing. Jobs are served FCFS until their service time exceeds a threshold. Then the job is preempted and placed at the end of the queue. The algorithm insures that the processor is equally shared by all its customers. It is an efficient algorithm providing that the overhead of managing the algorithm is not excessive.

STRF-shortest time remaining first. This optimal scheduling algorithm serves the customers with the smallest remaining outstanding service requirements. It is used often with drum devices. Over the long run, it tends to optimize the mean response time of all customers.

These scheduling algorithms are studied with respect to UP vs. MP processors. Many studies have proven limited differences in the performance of the various algorithms. Generally, FCFS is the easiest approach and has the least overhead.

What are the outputs of a model?

The outputs of a model are familiar performance metrics: response time, utilization, throughput, residency, average number of users in the system, to name a few. The outputs can be calculated by the queueing laws described earlier in this paper. Utilization means percent busy, system response time is the sum of residency time at each service center, R_k,

$R = \Sigma R_k$ where $k=1,..n$. Throughput is calculated through the forced flow law, and queue length can be derived from Little's law, or from the Response time law for TSO. By summing up all the constituent queue times you have the population of your entire system.

When a model has multiple service classes, they are parameterized on a class by class basis. All inputs and outputs are identical to that of the single class with the exception that metrics are assigned to that service class. Total response time for the system is weighted by throughput for each class.

Markovian Model

There are many popular mathematical models used in practical capacity management. The most prevalent of all queueing models is the so-called Markovian or memoryless model. The Markovian process, is also known as the M/M/1 Birth and Death process, where births equate to arrivals of customers (transactions or jobs) and deaths equate to customers exiting the system.

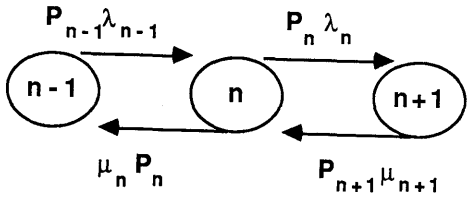
The Markovian model is used to accurately represent a computer system without including either too much detail or too little detail about past events which might influence the current state of the system. The first position of the M/M/1 model refers to interarrival time; the second to service time, and the third to the number of servers. Here we have exponential interarrival times and exponential service times with one server. (The M designates Markovian). In this model, the arriving transactions has 2 choices, it either receives service if the server is idle, or it must wait if there is a queue for service.

A queueing network can be represented by a Markovian process as follows. Each service center is represented by a Markovian state, or node, and there is a set of probabilities to identify transitions from one state to the next. When a customer completes service at the i^{th} node, he proceeds to the j^{th} node for service with probability P_{ij}. Each i^{th} node receives requests from outside the system as well as requests internal to the system.

A state transition diagram helps explain what Markovian process looks like. Each node represents a state Ei. The flow of requests into the node is

$E_n = (\lambda_{n-1})(P_{n-1}) + (\mu_{n+1})(P_{n+1})$ and the flow out is

$E_n = (\lambda_n)(P_n) + (\mu_n)(P_n)$. The flow out equals the flow in; the number of births equals the number of deaths.



Example

A system has one terminal at a branch sales office available 18 hours daily. The time spent at the terminal is on average 30 minutes-1/2 hour- for inquiries. There is only one terminal for 20 salesmen. The terminal is utilized 20x1/2 or 10. out of 18 hours per day. Let ρ be the utilization, then $\rho = 5/9$.

•What is the average wait time for the terminal?
W=expected service time/ $(1-\rho)$ = 30/(1-5/9)
= 67.5 minutes avg wait time

•What is the average number of salesmen in the queue?
The number in the queue $L_q = \rho^2/(1-\rho) =$
$(5/9)^2 / (1-5/9) = .7$ people waiting at any time

Two important practical capacity planning outgrowths of the M/M/1 model are:

(1) Expected time in the system for a customer = expected avg service time/ 1 - server's utilization

(2) Number of customers in the queue = server utilization^2/(1- utilization)

As the utilization increases, we see an increase in queue length and response time, similar to the traditional response time curve. As arrival rates of transactions linearly increase, residence time in the system grows exponentially. With utilization growth, a small increase in utilization corresponds to a significant increase in total time in the system. So overloaded systems operating at the 80-90% utilization level are much more sensitive to workload increases than those operating at the 60% busy level. If throughput is increased so that servers are busy for a shorter period of time, response time decreases.

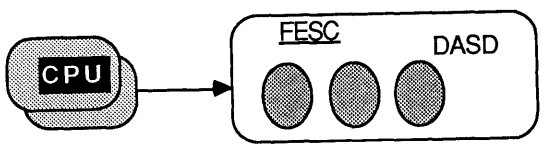
FESC's

We have covered techniques used to solve simple models so far. In particular, the Markovian technique is used to evaluate a small number of states. In large environments, where huge I/O and network systems are involved, there is a need to decompose the large super structure into more manageable systems. This process is known as *hierarchical* modeling and uses a concept called *FESC* or *Flow Equivalent Service Centers*. In constructing the FESC model, you first partition off the major subsystems, i.e. network, I/O subsystem and processors. Each component piece is analyzed as an individual entity, namely an FESC.

An FESC is a single, service center that exhibits delays and throughput characteristics on a large scale equivalent to the behavior of the detailed components of which it is comprised. After the decomposition process, you have a hierarchy of FESC's which are then evaluated from the lowest level of the hierarchy up. The result of this process, is the creation of many single service centers which are equivalent from a modeling perspective and smaller in scope than the original subsystems. The reduced model can be evaluated by a variety of mathematical techniques beyond the scope of this paper.

It is assumed that the subsystem is in equilibrium with the larger or macro system, of which it is a part. This occurs when the service centers in the subsystem can handle the influx of activity from outside,i.e. from the macro system.

FESC's are analyzed using "load dependent service centers" where the service rate depends on the actual number of customers in the queue. The FESC is considered to have a service rate equal to the throughput for all customers and classes that it serves, and the number of customers depends on the customers already in the macro system.

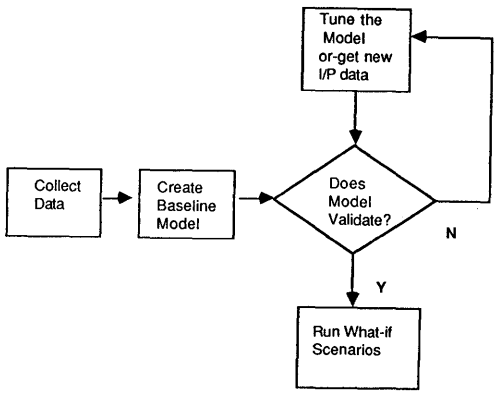


Practical Modeling

How do these theories and models relate to a pragmatic approach to modeling? To parameterize your current system, you must have on hand valid information about your customers, your service centers and your service requirements. The version of the operating system, the processor and I/O configuration, scheduling schemes, MPL levels and resource consumption by workload are some of the key inputs. Workloads must be characterized and grouped by function or by resource usage, i.e. by performance groups. Then the system and workload specific data is gathered through sampling techniques, i.e. through RMF. There are often other supplemental model inputs such as IMS transaction data from various vendor monitoring packages, since IMS workload metrics are not captured by RMF. The period of time chosen for the sample must be representative of the system's peak processing window, since most capacity studies are based upon maximum levels of activity.

Customer data (transactions, jobs) are decomposed into workloads of key concern to the organization. They correspond to performance groups and transaction groups for IMS and CICS. Service center data consists of the major queueing bottlenecks in the system, such as the processor, the DASD, or memory. Service requirements of each customer at each service center are identified through software monitors and samplers. System metrics such as CPU time, storage requirements, I/O activity by workloads are inputs to the model. However, measurement becomes complicated by the apportionment of system wide overhead attributable to MVS. This includes CPU burst activity, scheduling overhead, dispatching overhead and paging.

After the inputs are fed into the model, the model is validated for reasonableness. If the model validates, the modeled estimates do not differ dramatically from the measurements for an interval. Validating the model is not an easy process even though a small margin of error of 10-25% deviation is tolerated in the various measurements. When the model fails to validate, it is often due to some irregularity in the interval, such as the presence of an unusual job, lack of detail in the model definitions, or incorrect common system measurements such as poor distribution of paging overhead by workload. Validation implies either capturing a new set of more representative data, or changing metrics in the model, such as CPU burst, to alter the model to reflect the real system. Once the model validates, it can be used to pose what-if questions or to plan for different hardware and software scenarios.



Summary

This paper attempts to provide a general background covering analytic modeling fundamentals which are applicable to both formal and informal capacity planning. Hopefully, mysteries behind some of the commercially available packages begin to become clarified. In any aspect of performance and capacity management, a basic knowledge of queueing theory always adds to the decision making process.

Though real computer systems deviate from mathematical formulae, queueing theory serves as a foundation for performance estimates. If the analyst is willing to accept a broad range of accuracy, analytic modeling can be used to understand and explain performance results. This brief survey provides a glimpse into the behavior exhibited by many of the more complex queueing models implemented in capacity planning tools.

Bibliography

A. Allen, *Probability, Statistics and Queueing Theory,* IBM Systems Science Institute, CA (1978)

L. Kleinrock, *Queueing Systems Volume II,* Wiley, NY (1976)

Lazowska, Zahorjan, Graham, Sevcik, *Quantitative Systems Performance,*
Prentice Hall, NJ (1984)

Sauer, Chandy, *Computer Systems Performance Modeling,*
Prentice Hall, NJ(1981)

VM/HPO PERFORMANCE MEASUREMENTS

Mark S. Streger
Computer Sciences Corporation

ABSTRACT

Measuring the VM/HPO environment can be a challenging and difficult task. This tutorial will introduce those measures that quantify the different components of a VM/HPO system including CPU utilization, workload activity, user throughput and swapping. The focus will be on those measures that are both easily interpreted and are significant to performance analysis of a VM/HPO system.

1. INTRODUCTION

It is essential to realize that VM users are just like users of any other interactive time sharing system. Typical applications include fourth generation languages program development, information center packages and electronic office systems. Additional usage can include guest operating systems such as MVS or UNIX. VM systems typically provide the users with good responsiveness, but as I will describe, there are no good measures to inform either the performance analyst or the users of how the system is responding.

2. TRANSACTION TYPING

Performance analysts of other operating systems have developed over the years very concrete ideas of how to classify the workload and the service received. An excellent example is IMS response analysis with short, medium and long transaction average response times. Another example is TSO which prohibits a user from entering data at the terminal until the system is ready to accept it. This allows reporting service with command grouping. VM is much different. A VM system will allow a user to enter many commands in succession until the input buffer has filled up. The monitor data provided by the system does not characterize the workload, nor does it present data so that this can be accomplished. The monitor data does provide data that can be very useful, provided it is interpreted correctly.

3. THE VM MONITOR

VM provides the analyst with an integral data collection facility, the Monitor. This facility provides the analyst with two distinct types of data; interval data which quantifies activity over time and event data which is generated by specific activity. For most analysis the interval data is sufficient and the load generated is low enough to allow continuous collection. Event data may be necessary for analyzing obscure problems. Interval data has a default sampling interval of one minute which is typically used. The data itself originates from counters embedded throughout VM control blocks. It is worth noting that these counters are generally cumulative since the previous IPL, so that the analyst has to detect supurious data due to counter overflows. Monitor data needs to be reduced before it can be used for performance analysis with packages such as VMMAP or MICS/VM.

4. SYSTEM PERFORMANCE

Significant data describing system activity, subsystem utilization and numerous system counters can be obtained by collecting class 0 monitor data. This is interval driven data and the default one minute interval is typically used. Class 0 performance data provides an excellent overall view of the system so that current performance problems can be analyzed, as well as enabling the analyst to study the effect of changes upon the system. Some areas of interest include:

- Utilization: What the system was doing during the measurement interval.
- Workload : How many users were logged on the system and what portion of these were truly active.
- Storage : What the utilization was and what problems were affecting the storage manager.
- Swapping : How effective the swapping subsystem was. Are logically swapped users able to reclaim their pages.
- Throughput : How much user work was performed. Did users perform more work this week than last.
- Response : A comparitive measure of system responsiveness.

5. USER DATA

Collecting class 4 monitor data will generate a record for each logged on user during an interval. This data quantifies by user activity, resource consumption and delays incurred. Some of the measures need to be examined cautiously, as they can be skewed by a user who was logged on for only a few intervals. Some of the useful information provided by the user records include:

- Delays : What delays are most users experiencing. Are some users consistantly experiencing the same delays.
- CPU : Who are the largest CPU consumers.
- Storage : Which users have the large working sets.
- I/O : How much I/O the user performed.
- Response : No good measures at the user level.

6. I/O SUBSYSTEM

Collecting class 6 monitor data generates a wealth of data quantifying the I/O subsystem for disk and tape devices for each interval. This data allows analysis to be performed at the device, control unit and channel levels. Some of the useful information provided by the class 6 records include:

- Queueing : How often did a queue exist at this portion of the I/O path and what was the average queue depth.
- Utilization: How busy was the device, control unit or channel.
- I/O counts : How many I/O operations were issued to this path component.
- Response : No true measure, but can be approximated.

7. EVENT DATA

Collecting monitor event driven data can generate significant quantities of data depending on the event frequency. These classes are collected when the interval data does not provide sufficient granularity to analyze a situation. Some of the event data classes implemented provide information that is more suited to the VM developer when implementing an algorithm change, than to the performance analyst. The more interesting of the event data classes are:

Schedule : Exhaustive information is provided on the manipulation and activity to the dispatcher queues. This can be used to learn exactly how a user is serviced.

Seeks : This data provides information on DASD arm movement and is typically used as the basis for disk reorganizations.

Swap : This data class quantifies both logical and physical swapping. The information is presented on a user basis so that the effect of swapping on important users can be determined.

8. SUMMARY

This tutorial has examined the prime measurement data source of VM/HPO performance management and capacity planning. This discussion has focused on those mea ures that are important to basic performance analysis. The two distinct types of monitor data; event driven and interval driven have been contrasted, as well as the basic analysis that each type of data can provide. The basic and small set of measures discussed are sufficiently robust for basic VM performance analysis. As in depth knowledge of VM is developed, the additional measures that can be obtained can provide more insight to VM performance and aid in solving obscure problems.

CAPACITY PLANNING BY THE KISS METHOD

LARRY KAYSER
BGS SYSTEMS, INC.

Since the advent of formal capacity planning functions, more and more tools and techniques have been made available to the practitioner. Unfortunately more tools and techniques do not always lead to a successful process. Sometimes it is better to step back and look at the process in a more simplistic light. This tutorial will try to re-establish such a frame of reference.

A BIT OF HISTORY

In the early days of data processing, the environments were considerably less complicated. Capacity planning was geared to a few fundamental areas like the CPU and "when do I get my printout back." The rule of thumb in those early days might be likened to "if you don't care where you are, you ain't lost." Capacity planning, if attempted at all, most likely took some form of linear projection, based solely on CPU usage against time. The results were often like "The First Law of Evolving System Dynamics," "Once you open a can of worms, the only way to re-can them is to use a larger can."

A LITTLE CLOSER TO HOME

W ith the increase in computer technology and its uses, the demands placed on capacity planners have increased dramatically. The tools currently available to the capacity planner are far superior, but in my view at least one problem remains . . . **Understanding**.

Gone are the days of "Buy a bigger box." No longer is capacity planning just a function of CPU, but rather a complex interaction that crosses many boundaries. As more and more environments replace batch oriented systems with real-time, on-line systems, response time becomes the critical issue. Basic relationships exist within response time but they are a bit more complex than the sum of the parts.

A UNIQUE RELATIONSHIP

A relationship that I liken to a Rubics Cube. As with the Rubic's Cube, hindsight is great. With every twist of the cube you may be closer or further away from a successful solution. Capacity planning is like the cube. You know where you have been; you were balanced in the performance and cost of your DP environment. But how you get back to it is another question. The effects of changes now may be seen in problems further down the road. Especially if you rush into a solution.

WHAT IS CAPACITY PLANNING

So in a generic sense, what is capacity planning and how does one get started ? The objective of a capacity planning project is to maintain a fine balance between performance and cost effectiveness. This equilibrium is constantly stressed by several major factors.

Corporate management's view of DP is as a consumer of capital that may be better spent on other projects.

An increased DP exposure that comes from the use of on-line system.

More sophisticated users demanding:

- improved service
- better price performance
- new functionality

THE PROCESS

Capacity planning is a systematic process that:

- Forecasts User Workloads
- Determines Capacity Requirements to Support Those Workloads
- Manages Resources to Meet Defined Objectives

An observation from a number of years experience is that capacity planning and performance management are basically one and the same. A change in capacity can have an effect on the performance of an environment. And a change in demand can have an effect on capacity. Hence a "Catch-22". The effects from one must be weighed against the other to achieve the fine balance demanded in today's aggressive environments.

TECHNIQUES

There are numerous techniques for doing capacity planning/performance management. The basis of the process I present here is one of common sense. What I suggest are my ideas for having a reasonable chance at a successful capacity planning project.

First, my basic definitions.

Usable Capacity of a Resource -

The utilization beyond which the resource response is degraded to the point performance objectives are missed.

Performance Objectives/Service Requirements -

The ability to deliver a resource with a specific time frame.

GETTING STARTED

So how does one get started in capacity planning/performance management? My suggestions are based more on experience and common sense than on my formal training. They have been derived from a number of years of first hand exposure, both as a planner and as a vendor specializing in performance and capacity planning.

I believe there are eight essential points to a successful capacity planning process.

START SIMPLE

Practice the KISS principal and don't bite off more than you can handle. All that results from such attempts is frustration and failure. Tackle a piece at a time and use each as a building block to the next. The various pieces may include, among others :

- Tuning
- Costing and Charging
- Data Collection
- Workload Characterization
- Modeling

UNDERSTAND YOUR ENVIRONMENT

Each environment is unique and I promise you that if you are to be successful at capacity planning you will get to know your environment from CPU to terminal, from actuator to application. Get to know your users. Try to understand their problems and needs. Try to relate to them in terms they understand, not some obscure DP jargon. You will make your job a bit easier and possibly make some friends in the process. REMEMBER TO KISS.

PICK MANAGEABLE OBJECTIVES

Don't try to report everything in your system. Not only will you confuse yourself, but those you are reporting to as well. Try such things as CPU and response time by workload. Also track DASD usage in terms of volume. REMEMBER TO KISS.

TRACK YOUR OBJECTIVES

We all know that our projections are correct; but does reality reflect that as well? Remember, "If we don't care where we are, we can't possibly be lost." How well does reality track against your projections. This is why I stress KISS. If you are trying to manage to too many objectives when starting out, you might find that you are not meeting any of them.

REPORT ON YOUR OBJECTIVES

You don't need a book, when a memo will do. And conversely, when a major presentation is due, it is easier to have to deal with a few well managed objectives early on. Report only what is needed to make your environment understood. Basic CPU against current capacity, current response time against user requirements, projected CPU usage against capacity, and projected response time against user requirements. These are a few good ones to start with. REMEMBER TO KISS.

AVOID THE FLASHY

Too many times I have seen good management presentations get lost in the desire to be fancy and flashy. Somehow facts get lost in the process. KISSing is best. Avoid glitter and rely on substance. It works well. Spend the time working on the content and preparing alternatives for management.

PLAN PLAN PLAN

"A good plan today is better than a perfect plan tomorrow." Provide options and cost benefits to each. Most management I have dealt with prefer options. A few well thought out options are better than trying to razzle dazzle them. Try to anticipate what the future may hold and plan for alternatives, so you can reply quickly to management requests. But don't go overboard, REMEMBER TO KISS.

EDUCATION

Stay on top of technology. This business is one of the most dramatic in terms of changes, past or present. Be on top of options, hardware, software, or operational. It will make your job easier.

Diagram *#1* is the outline for a basic capacity planning process that was developed for a previous employer. I believe it represents a very sound structure to use as a guideline for establishing your own capacity planning process. Notice that it entails an understanding of all the areas I mentioned above and requires a great deal of interaction between the major components. It is also designed to be implemented in phases. By adhering to the KISS principle it becomes a manageable task that can and should evolve over time. Rome wasn't built in a day and neither is a good capacity planning project.

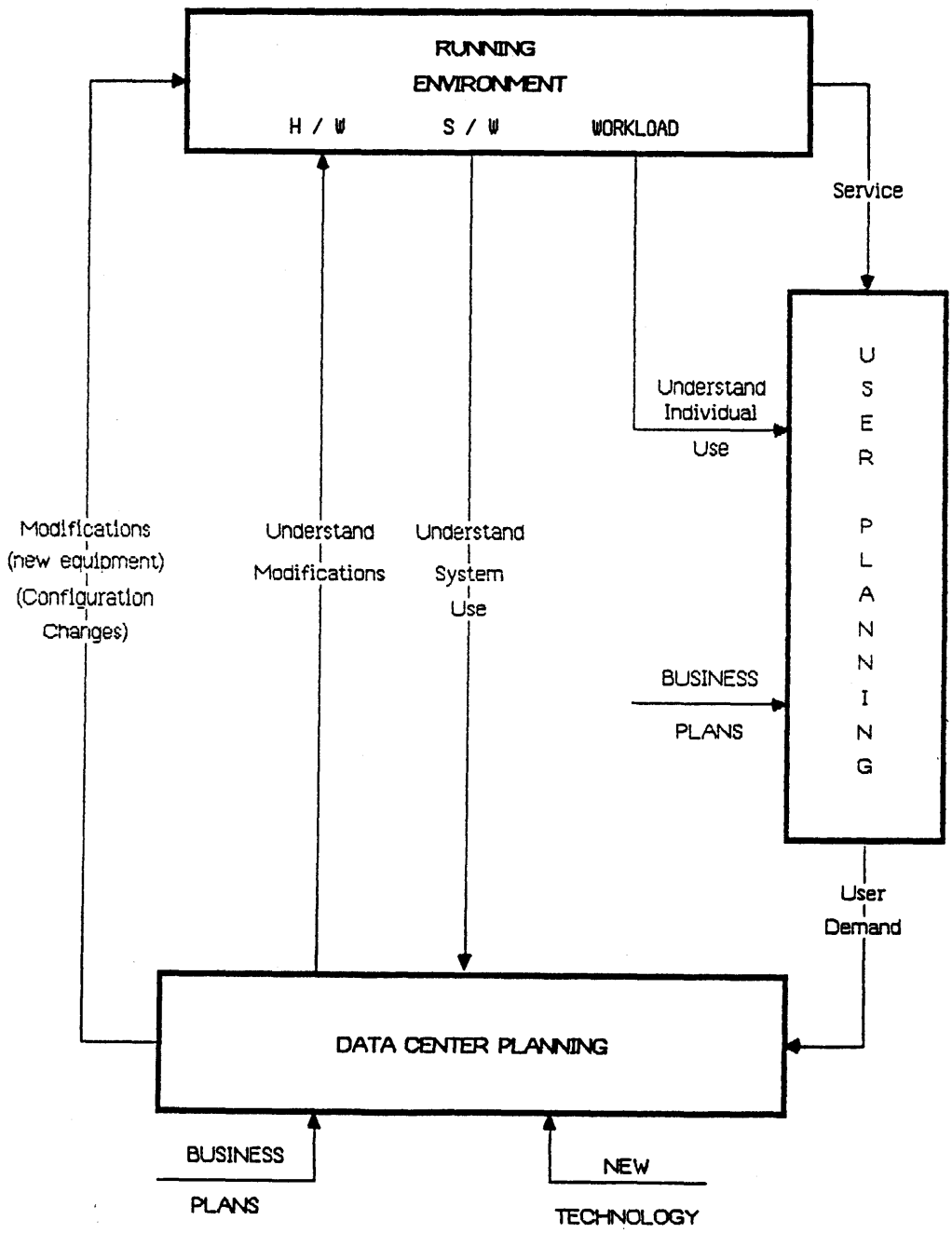
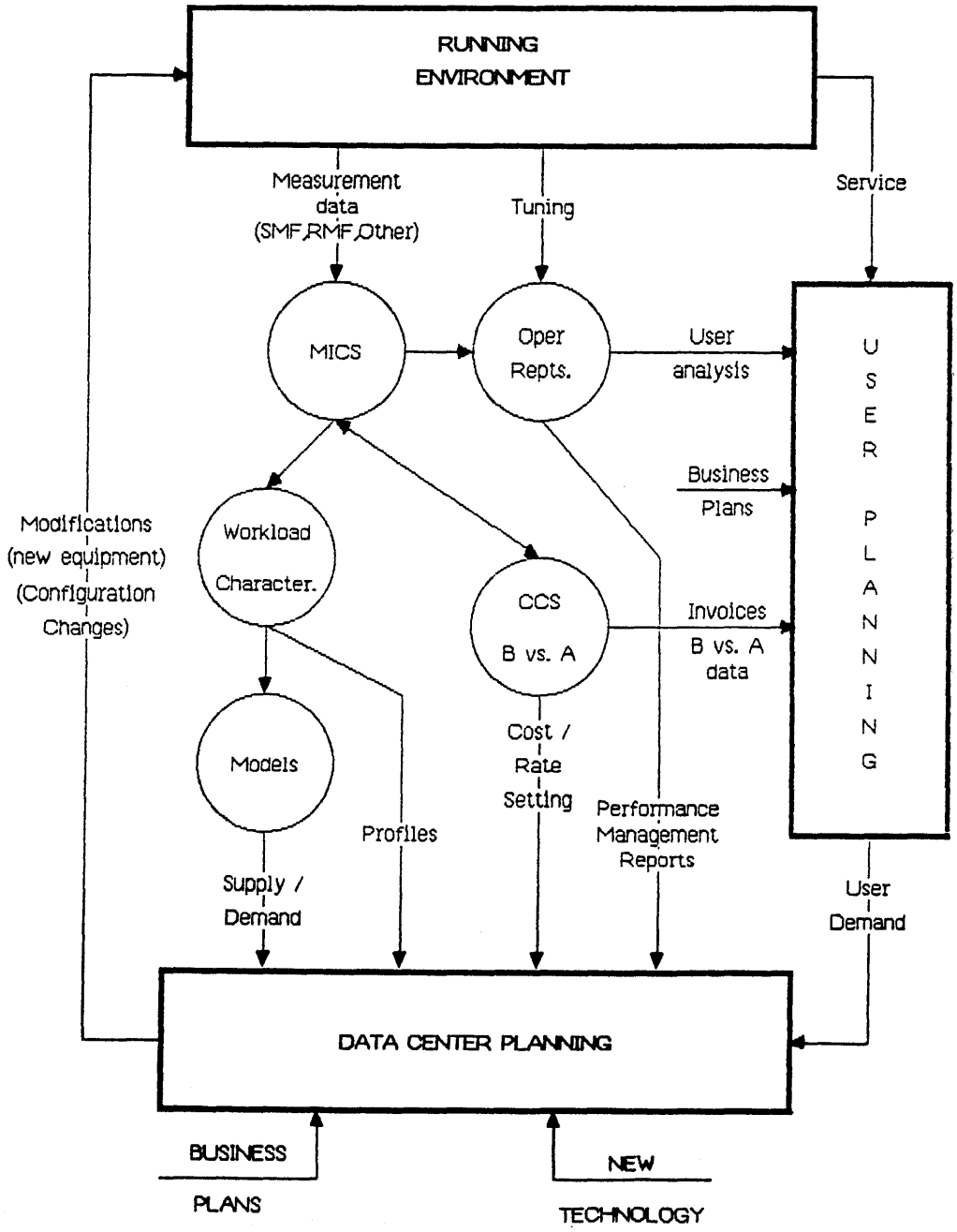


Diagram #2 is an example of an expanded process that was implemented at the same company. It will give you additional insight how the same capacity planning project grew. This particular one evolved into as many as seven full time people managing four major processing environments and several smaller support environments.



Finally, I would like to add a few footnotes. Capacity planners must be "a jack of all trades." They also need to be diplomats because they must deal with a number of areas within an organization, and corporate politics being what they are . . . enough said.

They need to be a promoter. They must understand and believe in their recommendations and be prepared to defend them. Many times management will question a recommendation, not so much in doubt of its value, but rather to test how well you have done your homework.

A capacity planner must be a technical ambassador, working between the various organizations promoting new technologies and better techniques to improve service and performance.

And last, for manager's of capacity planning projects, remember:

"*Nothing is impossible for the person who doesn't have to do it themselves*"

Give a new capacity planning function a chance to develop and don't push for the world when a small home in the country will do.

REPORTING AVAILABILITY AT THE JOB STEP LEVEL

Jim Scott
Virginia Department of Information Technology

ABSTRACT

Availability of a scheduled application can be defined as the percentage of scheduled time that the job step representing that application was executing. Using this definition, availability can be reported by a program whose input consists of a schedule table and step execution data from SMF or MICS. Relevant issues include these: What are the advantages and disadvantages of availability by job step? How should regular and irregular schedule variations be provided for? How can missing information (e.g., a missing step termination record) be accounted for? Is SMF interval accounting necessary? How can the availability report be presented online? How is the report used?

INTRODUCTION

The customers and managers of the Virginia Department of Information Technology (DIT) need daily information on the availability of scheduled applications. They use this information primarily as one measurement of how well DIT is doing its job. This information was originally calculated manually, but now the daily reports are generated quickly by a program. The program compares step execution start and stop times with the schedule. It then calculates and reports, for each application, the percentage of the scheduled time that the application was executing.

The key concepts are:

- The idea of basing availability on the execution of certain job steps, relative to a schedule;
- The particular method used by DIT to implement this idea.

These concepts are applicable to any installation that has scheduled applications implemented as job steps. It is advantageous, but not essential, that the schedules be reasonably stable.

AVAILABILITY BY JOB STEP

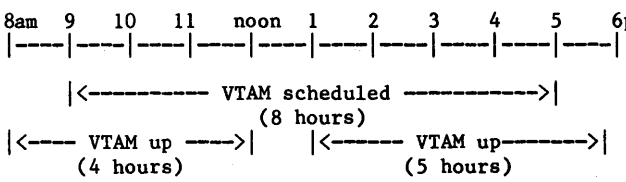
An application can be defined as a job that is scheduled to be up during certain periods of the day. It provides a service, usually in the form of online access to data, which is supposed to be available to customers in accordance with an announced schedule. When the application is up, the service is available.

For example, here are some of DIT's applications:

- JES2
- VTAM
- IMS control region
- CICS (two)
- Adabas (several)
- Division of Motor Vehicles (DMV) driver inquiry (daytime and nighttime versions)
- DMV vehicle update
- Legislature Quick Draft

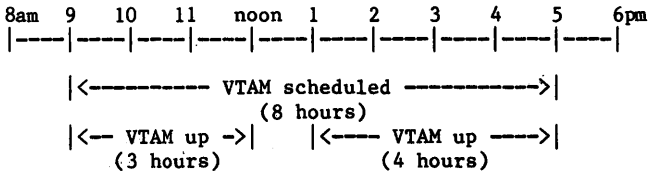
The job that supports an application may have several steps. There may be auxiliary steps that set up access to the data base, for example, or that do some kind of clean-up at the end of the job. Invariably, however, there is one main step that actually provides the scheduled service. Typically, the auxiliary steps run quickly, whereas the main step runs for hours and doesn't stop executing until operator action or a mishap causes it to stop. It's reasonable to equate availability of an application with the execution of the main job step for that application.

Thus, we say that an application is up between the time when its main step starts execution and the time when the main step ends execution. But in reporting availability, we only care about those periods when the application is **scheduled** to be up. It would be misleading to include execution time outside the schedule when calculating availability. Therefore, execution time outside the schedule is excluded. Consider the following example:



This diagram indicates that VTAM was scheduled to be up for a period of eight hours, from 9:00 a.m. to 5:00 p.m. VTAM actually came up an hour early at 8:00 a.m., and came down (because of some problem) at noon. It was down for an hour until the problem was fixed. Then it came up again at 1:00 p.m. and wasn't brought down by the operator until 6:00 p.m.

For the purpose of calculating availability, the diagram looks like this:



Here, the execution time outside the schedule has been ignored. The fact that VTAM was up early and stayed up an hour late may have been useful to some customers, and may even have compensated somewhat for the fact that it was down for an hour at noon. But there is no good way to measure that usefulness. The best we can do is to say that, outside the schedule, there is no such thing as availability.

In this example, then, the availability of VTAM is

$$\frac{\text{7 hours execution time}}{\text{8 hours scheduled time}} = 87.5\%$$

The advantages of measuring availability by job step are:

- It is well-defined:
- It is based on readily available data:
- It is a reasonable indicator of computer center performance.

The operating system provides reasonably accurate data for step start and stop times, and Operations policy provides official schedule start and stop times. Job step availability indicates how well the schedule is being adhered to.

The disadvantage of this method is that it may not measure "real" availability, in at least two senses:

- The service may not be really available the whole time the main step of an application is executing. For example, the main step may take a few minutes to do its initialization. In this case, the service isn't really available until a few minutes after the step begins execution.
- A much more fundamental problem is that job step availability is not the same as "end-user" availability. If an IMS message processing region is up and the IMS control region is down, then the user of the message processing region sees the service as being unavailable. The same would be true if both job steps were up but the front end processor that handled IMS customers' terminals was down. (Job step availability doesn't reflect network hardware availability at all.) In other words, job step availability is not a particularly accurate measure of availability as the customer perceives it. However, it is a good comparative measure of how well the computer center is handling one major aspect of its duty.

MANUAL REPORTING OF AVAILABILITY

In the past, availability was calculated by hand. Each morning, an operations analyst would go over an incident report filled in by operators over the previous day. For each application to be tracked, down times and up times would be noted and compared to the schedule. Availability percentages were calculated with a hand calculator.

There were two problems with this method:

- The times in the incident log were not particularly accurate. When a job or the system crashes, the operators' primary concern is to get it back up again. Recording the exact times involved would be of secondary importance.
- The manual effort was a waste of person-hours, and undoubtedly introduced occasional calculation errors.

INPUT DATA

DIT's availability report program requires three sets of input:

1. Execution data
2. A schedule table
3. Auxiliary tables

EXECUTION DATA

System Management Facilities (SMF) of the MVS operating system provides step execution data in the Type 30 (Common Address Space Work) record. DIT uses the MVS Integrated Control System (MICS) from Morino Associates, Inc., which makes SMF and other data available in a data base which is much easier to use than SMF files. In the MICS data base, step execution data are available in the Batch User Program Activity (BATPGM) file and the System Task Program Activity (BAT_ST) file.

DIT's fundamental cycle for reporting availability is a 24-hour period, midnight to midnight. Since we dump SMF data to tape once a day, at about 3:00 a.m., our SMF input would have to be two consecutive SMF tapes, covering from 3:00 a.m. the day before to 3:00 a.m. the day after the date to be processed. Each MICS "cycle" covers the same period as the SMF tape it's based on, so we use two consecutive cycles of each of the two MICS files mentioned above.

Even so, some steps run for days, so an application that's still up at the end of the day may not produce any step termination data until way after the 3:00 a.m. end of the input data. To account for this fact, we turned on SMF "interval accounting", with an interval of one hour. This causes each long-running step to produce step execution data in an SMF Type 30 record every hour starting one hour after the step begins execution. MICS then provides the same information. As long as SMF doesn't get dumped before 1:00 a.m., the input data can be assumed to reflect each step that was up as of midnight.

So the execution data to be used as input to an availability report can be SMF Type 30 records or MICS records: DIT uses MICS. Either way, SMF interval accounting is required, unless every scheduled application is brought down at least once a day.

SCHEDULE TABLE

A schedule table defines which applications are to be tracked; how to identify the execution data that corresponds to each application; and the schedule times for each application. It resides in a partitioned data set member maintained by Operations. Here's an example of a schedule table:

```
COMPONENT SCHEDULE
SMTWTFS DDMMMYY-DDMMMYY
 XXXXX  08JAN86 31DEC99
CICS PRD CICP2    DCSSIP   DFHSIP    0730 1715
CICS VEC CICP1    DCSSIP   DFHSIP    0730 1715
DRV INQD DMVPT02* T02DAY   DMVT02**  0650 2200
DRV INQN DMVT02NC T02NITE  DMVT02**  0000 0640 2205 2400
IMS CTL  IMSPCU   IEFPROC  DFSMVRC0  0615 1900
IMS MPP1 COMMPP1  COMMPP1  DFSRRC00  0620 1900
JES2     JES2     IEFPROC  HASJES20  0000 2400
LEG QD   LEGQDCU  QD       QDDRAFT   0735 2400
VEH UPDT DMVPT51C DMVT51** DMVT51**  0650 2200
VTAM     NET***** NET***** ISTINM01  0000 2400
```

The first three lines are used to identify the member and to specify to which dates and to which days of the week the schedule applies. In the rest of the table, each line is an entry containing the following items:

Name An eight-character name for the application: long enough to be recognizable, but short enough to save space in intermediate files

Job name The job name always associated with this application

Step name The name of the main step

Program name The program executed by the main step

Schedule period 1 Schedule start and stop times

Schedule period 2 Start and stop times for second schedule period, if any

The job name, step name, and program name fields define which execution data records represent a particular application. Note that wild-card characters are allowed in these fields.

In the example above, the application with name "DRV INQN" is scheduled to be up from midnight to 6:40 a.m. and from 10:05 p.m. to midnight (i.e., its actual schedule is 10:05 p.m. to 6:40 a.m.). It has job name DMVT02NC, step name T02NITE, and any program name beginning with DMVT02.

There can be several schedule tables with different member names. DIT maintains separate tables for:

- Sunday
- Monday
- Tuesday-Friday
- Saturday
- Holidays (two types)

AUXILIARY TABLES

The three auxiliary tables are the description table, the holiday table, and the window table. They're not essential to the concept of job step availability; they just facilitate the process. Like the schedule tables, they are partitioned data set members.

DESCRIPTION TABLE. The description table equates each short application name (as used in the schedule table and in intermediate files) with a longer application description, to be used in reports. Here's an example:

```
COMPONENT DESCRIPTIONS
SMTWTFS DDMMMYY-DDMMMYY
XXXXXXX 09JUL85 31DEC99
CICS PRD CICS PRODUCTION
CICS VEC CICS - VEC
DRV INQD DMV DRIVER INQUIRY (DAY)
DRV INQN DMV DRIVER INQUIRY (NIGHT)
IMS CTL  IMS CONTROL REGION
IMS MPP1 IMS MSG PROC REGION 1
JES2     JES2
LEG QD   LEGISLATURE QUICK DRAFT
VEH UPDT DMV VEHICLE UPDATE
VTAM     VTAM
```

HOLIDAY TABLE. The holiday table identifies which dates are holidays, so selection of a schedule table can be done without manual intervention:

```
HOLIDAY TABLE
SMTWTFS DDMMMYY-DDMMMYY
XXXXXXX 01JAN85 31DEC88
2 %% 01 01 NEW YEAR'S DAY
2 87 01 02 NEW YEAR'S EXTRA DAY 87
1 85 10 14 COLUMBUS DAY 85
1 86 10 13 COLUMBUS DAY 86
1 87 10 12 COLUMBUS DAY 87
1 88 10 10 COLUMBUS DAY 88
```

The items in each entry, from left to right are:

Type The type of holiday, 1 or 2. A type 1 holiday is a minor holiday on which many businesses stay open. A type 2 holiday is a major holiday. DIT has a different schedule for each type of holiday.

Year The year that this entry applies to, or '%%' if the holiday has the same date every year.

Month The month of the holiday

Day The day of the month

Description The name of the holiday

WINDOW TABLE. A window table defines two time periods:

- The start and stop times of a scheduled maintenance window, if any; and
- The start and stop times of the period of the day, if any, considered to be "prime time".

A **maintenance window** is a period of a few hours reserved for preventive maintenance, repairs, or testing, on a particular day of the week. **By definition**, nothing is scheduled during a maintenance window, regardless of what the schedule table says. Any execution time and any schedule time that would overlap a maintenance window are ignored. Of course, windows could be reflected in the schedule tables, but then more schedule tables would have to be maintained.

The concept of **prime time** is used to report availability, not only over a 24-hour period, but also for the prime time period of the day. Prime time availability ignores any execution time or schedule time outside of prime time. Prime time availability can be useful, for example, in the following way: Suppose that on a particular day the availability of JES2 was 84%, but prime time availability of JES2 was 98%. Management may want to say that this was not as bad as a day when availability of JES2 was 90% and prime time availability was 92%.

There are two window tables, one for regular work days and one for holidays and weekends. Here are both tables:

```
SCHEDULED WINDOWS - MONDAY THROUGH FRIDAY, NON-HOLIDAYS
SMTWTFS DDMMMYY-DDMMMYY  WINDOW    PRIMETIME
 XXX X  22JAN86 31DEC99            0700 1730
    X   22JAN86 31DEC99 0250 0505  0700 1730

SCHEDULED WINDOWS - HOLIDAYS AND WEEKENDS
SMTWTFS DDMMMYY-DDMMMYY  WINDOW    PRIMETIME
X       22JAN86 31DEC99  0750 1205
 XXX X  22JAN86 31DEC99
    X   22JAN86 31DEC99  0250 0505
      X 22JAN86 31DEC99
```

One of these tables is selected, based on the day of the week and on whether the day is a holiday (according to the holiday table). Then, an entry from the table is selected, based again on the day of the week (and possibly on the date range). For example, on Thursday, December 10, 1987, which is not a holiday, there is a maintenance window from 2:50 a.m. to 5:05 a.m., and prime time is defined as 7:00 a.m. to 5:30 p.m.

Note that there is no prime time on weekends and holidays. On these days, prime time availability is undefined.

OVERVIEW OF PROGRAM

The program that calculates and reports availability is written in SAS (from SAS Institute, Inc.). It's executed each day by a scheduled job that runs after SMF has been dumped and MICS has been updated. The operations analyst who is responsible for the job must make sure that the schedule tables and auxiliary tables are kept up to date. When the job is submitted, a date parameter may be used to specify the date to be processed; the default is yesterday (the day before the date the job is being executed).

The program, knowing the date to be processed, then goes to work:

1. The day of the week is determined.

2. The holiday table is used to determine whether the day is a holiday, and if so, which type.

3. The proper schedule table and window table are selected and read in. The description table is also read in.

4. The MICS cycle numbers (generations) to be used are determined. If the date to be processed is yesterday, the cycle numbers used are 1 (the latest day's worth of data) and 2 (the next-to-latest). If an earlier date is being processed, the cycle numbers will be higher (for older data). This may occur if a previous run needs to be rerun. See "Handling Bad Data", below. Since a limited number of cycles of MICS detail data are kept, there is a limit to how far back a rerun can be done.

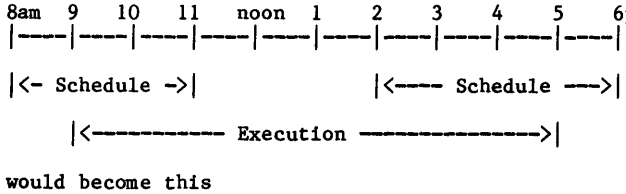
5. The execution records are read in and summarized to produce one record per job step execution (by combining all the interval records for each step).

6. Execution records are discarded for steps that did not overlap the date being processed, and for steps that lasted less than one minute.

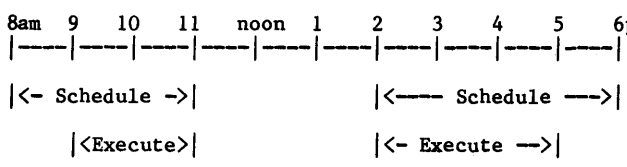
7. The remaining execution records are merged with the schedule table, based on job name, step name, and program name (and taking any wild-card characters into account). This process eliminates most of the execution data, leaving only the records for job steps whose availability is to be reported, i.e., those mentioned in the schedule table. It also associates a set of schedule times with each surviving execution record. A schedule entry with no matching execution data becomes a record of an application with no execution time.

8. Execution time outside the process date is excluded. For example, a step which started the day before and ran until 4:00 a.m. of the date being processed is considered to have run from midnight (the beginning of the day) to 4:00 a.m. (A SAS time constant of '00:00'T indicates midnight at the beginning of the day, and '24:00'T indicates midnight at the end of the day.)

9. Any execution which overlaps two scheduled periods is broken up into two executions by excluding the non-scheduled period in between. For example, this



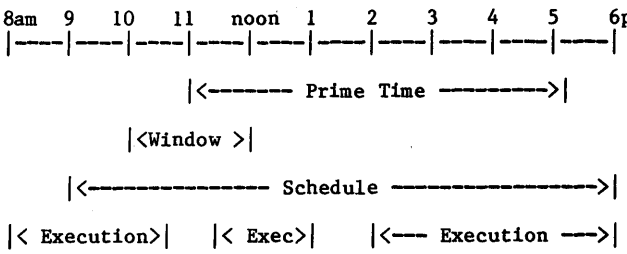
would become this



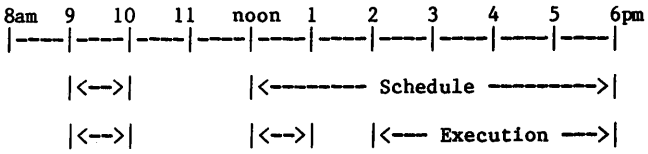
10. Overlapping executions are de-overlapped. The start time of one is moved later, or the end time of the other is moved earlier, or an execution totally contained by another is dropped entirely, so that the overlap is removed. An arbitrary minimum down time of one minute is imposed.

(There are various ways that two executions with the same job name can overlap. A batch job and a started task with the same name can execute at the same time. Also, the use of wild-card characters in the job name field of the schedule table can result in apparent duplicate execution.)

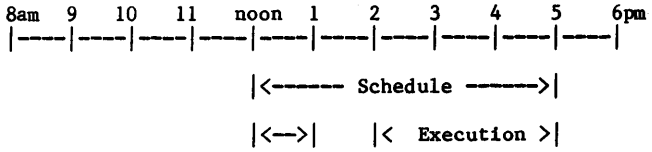
11. The data are "normalized". Schedule time during a maintenance window is excluded. Execution time outside the schedule is excluded. An execution which spans a maintenance window (probably meaning that the CE didn't take the machine down after all) is broken into two pieces, as above. A second set of data is also kept for prime time availability, by excluding all execution and schedule time outside prime time (if any). For example, suppose the input data for an application looked like this:



For the purposes of 24-hour availability, the schedule and execution data would be converted to this:



For prime time availability, they would look like this:



12. Availability is calculated for each application by computing the ratio of total execution time to total schedule time. For the application in the example above, availability is

$$\frac{\text{6 hours execution time}}{\text{7 hours scheduled time}} = 85.7\ \%$$

13. Likewise, prime time availability is calculated for each application by computing the ratio of total prime time execution time to total prime time schedule time. In the example above, prime time availability is

$$\frac{\text{4 hours execution time}}{\text{5 hours scheduled time}} = 80.0\ \%$$

14. The number of "hits" or interrupts is also calculated. A hit is an incident in which an application's job step does one of the following:

    - Comes up later than scheduled;
    - Goes down earlier than scheduled;
    - Goes down and comes back up during the schedule;

o Isn't up at all during a schedule period.

In the example above, the application had **one** hit, the down from 1:00 p.m. to 2:00 p.m. The down at 11:00 a.m. was completely within the window, and therefore was not counted as a hit because it was considered to have been outside the schedule.

This definition of a hit is probably the most practical one for computation by a program, but it isn't always fair. For example, a nighttime application may go down at 11:30 p.m. and come back up at 12:30 a.m.; this one down would be reported as two hits, one on each day.

15. The daily availability report is printed, and stored in a partitioned data set for online access.

16. The month-to-date availability report is calculated, reported, and stored. For each application, the day's total execution time is added to the total execution time for the preceding days in the calendar month, and likewise for schedule time. Then the month-to-date availability is calculated by dividing month-to-date execution time by month-to-date schedule time. The month-to-date data are stored in a SAS data base for use in the next day's calculations.

    After the month ends, the month-to-date report for the last day of the month is then considered the "monthly report" for that month.

17. If the date being processed is a Saturday, the weekly availability report is calculated, reported, and stored. The calculations are similar to those for the month-to-date report, but the weekly period is always Sunday to Saturday.

THE REPORT

The daily availability report looks like this:

DEPARTMENT OF INFORMATION TECHNOLOGY
MVS SYSTEM
DAILY AVAILABILITY REPORT
FOR THE DATE OF: WEDNESDAY, DECEMBER 9, 1987
COMPUTED ON: 10DEC87:08:41:03

APPLICATION DESCRIPTION	EXECUTION TIME	SCHEDULE TIME	AVAIL PCT	PRIME AVAIL	# OF HITS
CICS - VEC	9:45:00	9:45:00	100.00	100.00	0
CICS PRODUCTION	9:45:00	9:45:00	100.00	100.00	0
DMV DRIV INQ (DAY)	12:53:50	15:10:00	85.04	100.00	3
DMV DRIV INQ (NITE)	8:33:17	8:35:00	99.67	.	1
DMV VEHICLE UPDATE	12:54:26	15:10:00	85.10	100.00	3
IMS CONTROL REGION	12:45:00	12:45:00	100.00	100.00	0
IMS MSG PROC REG 1	12:30:56	12:40:00	98.81	100.00	1
JES2	23:39:15	24:00:00	98.56	100.00	1
LEGISLAT QUICKDRAFT	13:59:07	16:25:00	85.19	98.38	3
VTAM	23:25:25	24:00:00	97.60	100.00	1

The month-to-date and weekly reports are similar.

"Execution Time" is the total step execution time within the schedule, for the whole day (or month to date, or week). "Schedule Time" is the total time the application was scheduled. "Avail PCT" is the (24-hour) availability, as a percentage ratio of execution time to scheduled time. "Prime Avail" is the prime time availability percentage; prime time execution time and scheduled time are not included in the report. "# of Hits" is the number of hits, or interrupts, as defined earlier.

The reports are stored as ready-to-browse partitioned data set members, and are accessed through a TSO ISPF CLIST. The Library Access and Dialog Management services of ISPF are used to display a list of members from which the viewer can choose a report to browse. The member list includes all daily reports for the current and previous months, all month-to-date reports for the current month, all weekly reports for the current and previous months, and all previous monthly reports (last month-to-date report for each month).

HANDLING BAD DATA

All sorts of bad data can get into the availability report, either through bad input data or a program bug. These can generally be found in one of two ways: either the program detects a problem and issues a warning or error message, or the operations analyst who reviews the daily report notices that the report looks wrong. Such problems can be corrected by doing a rerun.

KINDS OF BAD INPUT DATA

Here are some examples of bad input data:

BAD ENTRY IN SCHEDULE TABLE. Either a data entry error or a miscommunication concerning the day's schedule could result in the schedule table having one or more bad entries. The resulting report would, of course, also be wrong.

WRONG SCHEDULE TABLE. The operations analyst could have the wrong member names on the schedule tables, in such a way that the program runs with the wrong table.

MISSING ENTRY IN SCHEDULE TABLE. A missing entry in the schedule table would make it look as if an application was not scheduled.

OUT-OF-DATE HOLIDAY TABLE. Errors in the holiday table could cause the program to select the wrong schedule table.

BAD ENTRY IN WINDOW TABLE. A bad entry in the window table could misstate whether a maintenance window and/or prime time was in effect.

AVAILABILITY REPORT JOB SUBMITTED BEFORE SMF OR MICS DATA UPDATED. If the report were generated before the input data were updated, it would use old data, resulting in a report showing zero or very low availability.

DAMAGED MONTH-TO-DATE OR WEEKLY DATA FROM PREVIOUS DAYS. If month-to-date or weekly data from previous days were not correct, the latest month-to-date or weekly report would also be incorrect, even though the latest daily report might be all right.

MISSING STEP TERMINATION RECORD. The SMF or MICS records representing a job step execution may show either of the following two conditions:

Completed step The last record for the step is an end-of-step record:

Incomplete step The last record for the step is an interval record.

The latter case is often okay; it may just mean that the step was still running when SMF was dumped. But if the last timestamp for the step was before midnight and before the end of the schedule period for the application, then it isn't known whether the step actually ended before or after it was scheduled to end. This problem can be caused when an initiator terminates abnormally.

MISSING INTERVAL RECORDS. It's possible for a step not to issue interval records even though SMF interval accounting is turned on. This has been a continuing problem for JES2; at IPL time, it is a matter of luck

whether JES2 or SMF gets completely started first. If JES2 is first, it doesn't get the signal from SMF to produce interval records. When a job step that does not produce interval records runs past the SMF dump time (as JES2 often does), MICS has no record that the step has been running.

CORRECTING BAD DATA BY DOING A RERUN

The effect of any of these kinds of bad input data, or of a program bug, can be corrected, assuming that the correct input data can be determined or that the bug has been fixed. A "full" rerun can be done for any past date for which the original execution data still exist. DIT keeps 10 cycles of MICS detail data; since two consecutive cycles are required as input, we can rerun any date from yesterday back to nine days ago.

We also allow a "partial" rerun, in conjunction with **manual** correction of the execution data. Manual correction is required when a missing step termination record and/or missing interval records obscure whether (or when) a step ran and ended. We allow two kinds of manual correction of execution data:

- Changing a job step's execution end time
- Adding a new job step execution

Of course, manual correction depends on the conscientiousness of the operations analyst, who must determine from the system log or other sources of information what the execution data should be. The partial rerun checks the manual corrections for consistency with other data, but this check doesn't eliminate the need for human carefulness.

In practice, DIT uses manual correction for JES2 almost every day. Manual correction for other applications, and full reruns to correct other kinds of bad input data or program bugs, are considerably less frequent.

USE OF THE EFFORT

The availability report is used by DIT managers, other DIT employees, and some customers. They use it in the following ways:

- As one measure of how well DIT is doing its job, while recognizing that availability is also affected by factors outside DIT;
- As an indicator of problems that may need further investigation, using other tools (such as the Incident Report or the system log);
- As a factor in evaluating the performance of DIT managers.

SUMMARY

Job step availability is based on the idea that an application is up when its main job step is executing. Availability is then the percentage of the scheduled time that the step was executing. Although this measure of availability doesn't always reflect whether the customer has access to the scheduled service, it's still a reasonable measure of data center performance. Calculation of job step availability is more complicated than one might think, but it is quite practical.

WHAT MODELS DO AND WHAT MODELS DON'T

Allan I. Levy
International Systems Services Company
New York, NY

This tutorial is intended for both intermediate and advanced level performance and capacity analysts. The tutorial presents details of the practical application of queueing network modeling technology to computer capacity planning. The tutorial explores realistically what can be expected of these models. It also explores the limitations of these models. Major emphasis is placed on identifying the modeling primitives required to address specific capacity planning and computer performance issues. Emphasis is also placed on the formulation and architecture of a model, modeling types, model input requirements, model modification analysis, and model validation requirements. Extensive use will be made of practical case studies drawn from the industry.

TUTORIAL OUTLINE

1. Models of Computer Performance

1.1 Rule-based models
1.2 Utilization models
1.3 Analytic models
1.4 Simulation models
1.5 Hybrid models
1.6 Hierarchical models
1.7 Benchmark models

2. Queueing Network Models of Computer Performance

2.1 Transaction processing models
2.2 Timesharing models
2.3 Batch processing models

3. Using the Model

3.1 Choosing the model to fit the problem
3.2 Model architecture
3.3 Model structure
3.4 Obtaining the model input
3.5 What to do when required input is not available
3.6 Using the model output
3.7 Mis-using the model output
3.8 How to validate a model
3.9 How not to validate a model
3.10 When is a model considered validated

4. Model Modification Analysis

4.1 How to modify model parameters
4.2 Tricks of the modeling trade
4.3 Modeling problem areas and their resolution
4.4 Models of batch precedence
4.5 Models of DASD
4.6 Models of CPU serialization
4.7 Models of shared DASD
4.8 Models of the I/O subsystem
4.9 Models of CPU-I/O overlap
4.10 Models of fork and join processing

5. Pragmatic Case Studies

5.1 Modeling economics in a credit card retail application
5.2 Modeling remote data base updates in a large bank
5.3 Modeling on-line transaction processing in a distributed environment
5.4 Other cases, including those requested by tutorial participants

ACCOUNTING FOR WORKLOAD VARIATION IN INTERVAL SELECTION AND ANALYSIS

Brian Williams
American Express
Travelers Cheque Operations Center

ABSTRACT

Selecting evaluation intervals which are representative of critical performance periods is important. This is particularly true if a period is to be used as the basis for an analytic model from which future performance projections will be made. Using a "peak" period based on CPU utilization alone may not be adequate in selecting a "representative" period of critical performance. This paper outlines a method of identifying periods of critical performance taking into consideration important factors such as workload mix, service levels, CPU utilization, and workload volumes. First, critical processing periods are defined. Then a process is described that combines all important factors into a single **indicator** that can be used to determine how well a particular interval represents a critical performance period. Uses and possible implementation strategies of this single indicator value are discussed along with the potential benefits of this methodology.

INTRODUCTION

Critical performance periods have traditionally been defined as "peak" intervals (usually hours). Peak intervals are determined by analyzing daily, monthly, and seasonal patterns to identify cycles and periods of workload variation. Peak periods are then defined as intervals in the cycles that have the highest workloads. Performance tuning and capacity planning are usually oriented towards the peak periods or "critical" performance periods because these periods have the highest demand on resources.

When selecting a peak period for analysis, it is important to know whether this peak period is representative of processing done by the configuration when workload variation is taken into consideration. Determining if a critical performance period is representative of processing done on the system will require a definition of "representative". A number of important factors that relate to workload variations must be considered in the definition.

A DEFINITION OF CRITICAL PERFORMANCE PERIODS SHOULD CONSIDER SERVICE LEVELS

Satisfying service objectives under every condition is not economically practical and may not be feasible. Therefore, a company must determine in what situations service levels may not be satisfied. One approach is to set a percent of the time that service levels may not be satisfied. For example, ninety percent or more of the time all service levels will be satisfied. Then only ten percent of the time (or less) service levels may not be satisfied.

The time period must also be defined. That is, ten percent or less of *what* periods will service levels not be satisfied. A company must determine what continuous periods of time can the users function while service levels are not satisfied. This will depend on the nature of the business, the type of processing provided, and the service objectives. For example, a system providing batch processing for randomly submitted jobs may be able to sustain unsatisfactory service levels for hours and still not significantly impact the user community. Selecting longer intervals for predominantly batch work will allow a better description due to the nature of longer running batch jobs. However, a system providing online services to support sales and customer service during a peak season of the year may lose substantial revenue if unsatisfactory service levels are sustained for more than one hour. Therefore, with critical online workloads short periods would be more appropriate.

For example, a company may determine that ninety percent or more of all one hour periods must satisfy service objectives. The company may also determine that the time period may need to be confined to a particular part of the year or a particular season (i.e. peak season). The critical performance periods must be part of the periods for which service levels must be satisfied. For example, ninety percent of the one hour periods during peak season are candidates from which representative critical performance periods can be selected.

The most critical performance periods are those periods which border the intervals for which service levels are not expected to be satisfied. For the purposes of this paper, critical performance periods will be referred to as ninetieth percentile performance periods. Ninetieth percentile periods are those borderline periods which separate the ninety percent or more periods in which service levels are expected to be satisfied from those ten percent or fewer periods in which service is not expected

to be satisfied. Based on the discretion of a company the ninetieth percentile or any other percentile could be selected.

The remainder of this paper assume the following critical performance period definition:

> Critical performance periods will be any continuous four hour ninetieth percentile periods during any quarter of a standard calendar year.

This definition uses a less traditional four hour period and allows selection criteria to be established quarterly. Using this criteria over a specified period such as a calendar year quarter will account for all daily, weekly, and monthly cyclical patterns and/or peaks. If applied over a one year period, seasonal patterns and/or peaks are also accounted for. If one month is analyzed, daily and weekly cyclical patterns are accounted for but monthly and seasonal patterns may not be considered. The interval of data to be analyzed should depend on the types of cycles expected in the processing schedules. Analyzing longer periods will consume more resources.

Where there is a large seasonal variation in workloads, it is likely that all periods in a seasonally slow quarter would meet service objectives. However, by definition the periods representing ninetieth percentile workload volumes are the critical periods. If quarterly data is analyzed and seasonal variations in workload exist, seasonal adjustment would be necessary for predicting peak season performance.

THERE IS NOT ONE UNIQUE FACTOR THAT CAN IDENTIFY CRITICAL PERIODS

Once critical performance periods have been defined, criteria must be established for identifying these periods. A technique must be developed to identifying those periods for which service levels are not expected be satisfied. One alternative is to use CPU utilization. Critical performance periods could be selected from periods where CPU utilization was the same or lower ninety percent of the time. However, consider the data collected from an IMS system in **TABLE I.**

Each interval in the table is bordering on the ninetieth percentile period for CPU utilization. The range of CPU utilization is only 3%, but the range in workload volumes is significant. Which is the the most representative interval? The interval selected will have a large impact on future IMS response time projections. The use of CPU utilization exclusively to select a representative period is clearly insufficient.

If CPU utilization is not sufficient to determine if an interval is representative of a ninetieth percentile period, perhaps workload volumes should be considered. The intervals in **TABLE II** are borderline ninetieth percentile intervals based on IMS transaction volumes.

The difference in IMS transactions for each interval is insignificant. However, the range in CPU utilization and the range in BMP data base DL/1 calls is very significant. The IMS transaction response time is more consistent but the highest response time is twice as large as the lowest response time which also would significantly impact future performance projections. Using only workload volumes from one workload is also not sufficient to indicate that a period is representative of

TABLE I

INTERVAL	AVERAGE CPU UTILIZATION	NUMBER OF IMS TRANSACTIONS	IMS TRANSACTION RESPONSE TIME	BMP DATA BASE DL/1 CALLS
1	77.66%	16,826	3.66 SEC	2,167,797
2	77.36%	17,453	0.28 SEC	709,770
3	76.40%	22,278	0.97 SEC	1,737,203
4	76.35%	19,964	0.41 SEC	1,098,527
5	75.68%	21,804	0.53 SEC	1,916,961
6	74.84%	17,488	1.59 SEC	2,221,426

Figure 1. Periods with similar CPU utilization: Although CPU utilization is similar, IMS online transactions, online response times, and BMP data base calls are significantly different. This indicates that using only CPU utilization as an indicator of performance levels for selected period is inadequate.

TABLE II

INTERVAL	NUMBER OF IMS TRANSACTIONS	AVERAGE CPU UTILIZATION	IMS TRANSACTION RESPONSE TIME	BMP DATA BASE DL/1 CALLS
1	23,707	64.6%	0.35 SEC	1,445,604
2	23,640	56.4%	0.29 SEC	1,672,766
3	23,551	47.7%	0.49 SEC	170,773
4	23,525	49.7%	0.31 SEC	147,937
5	23,516	29.3%	0.24 SEC	44,280
6	23,355	52.9%	0.28 SEC	2,513,754

Figure 2. Periods with similar IMS online transaction volumes: These periods have similar IMS online transaction volumes. However, it can be seen by the CPU utilization, IMS online response time and BMP data base calls that these periods are also very different. Many factors must be used to determine the similarity of different periods when there is variation in workloads.

ninetieth percentile periods. Furthermore, service levels (IMS response times) vary due to the change in mix within the workloads (IMS online transaction workload). Therefore, service levels must also be considered.

It becomes apparent that the levels of several measurement factors must be considered to determining if an interval is representative of critical performance periods.

FACTORS USED TO IDENTIFY INTERVALS WILL BE DIFFERENT FOR EACH COMPANY

The factors that should be considered in identifying intervals representative of ninetieth percentile periods will depend on the nature of the business, the workloads, and service levels. Workload volumes and service levels of the major workloads in the configuration should be used in identifying ninetieth percentile periods. For capacity planning purposes, since a major objective is to plan for adequate processor resource, CPU utilization should also be considered.

For example, consider the following system. The primary production work involves IMS online and BMP processing with some O/S batch processing. Based on these workloads, the following factors might be used to determine ninetieth percentile periods:

IMS online transactions (for each different class of transactions)

IMS online transaction response time

BMP data base DL/1 calls (for each different class of BMP's)

BMP multiprogramming level

Production O/S batch job-steps per hour

CPU utilization

Let IMS transactions be classified by resource consumption with class 1 consuming the least resources and each class after that requiring more resources per transaction. Let BMP's be classified in a similar manner with class 1 requiring the least amount of resources per data base call.

Using the definition previously established for critical performance periods (four hour periods over a quarter) and the factors above, the critical performance periods must be identified by analyzing fifteen measurements for every four hour period in the quarter. That is, using the fifteen measurements from each four hour period, ten percent of the four hour periods must be identified as periods where service levels would not be *expected* to be satisfied (although service objectives may have been satisfied during those periods). Furthermore, periods which border the ninetieth percentile based on these fifteen measurements must be identified as the critical performance periods. To accomplish this, a definition of the *ninetieth percentile borderline* must be made in terms of these fifteen measurements.

A Ninetieth Percentile Borderline is Defined by the Identification Factors

A ninetieth percentile borderline period should be a period which represents ninetieth percentile CPU utilization. By considering the workload volumes, an appropriate workload "mix" will be determined as criteria for a ninetieth percentile borderline. In considering workload mix, is the "normal" workload mix the

desirable mix for a ninetieth percentile period? This is a decision made at the discretion of each company. Past experience would indicate that for a ninetieth percentile borderline period, each workload volume should be at a ninetieth percentile level. Furthermore, *each* factor used in the evaluation should be at a ninetieth percentile level.

Therefore, the ninetieth percentile borderline is defined as a period where all factors are at ninetieth percentile levels. It is reasonable to believe that if the CPU utilization is at the ninetieth percentile level, that workload volumes should also be at ninetieth percentile levels.

Ninetieth Percentile Periods Will Probably Not Have "Typical" Workload Mixes

It may be of some concern that the workload mix is *typical* if all workload volumes are at ninetieth percentile levels. However, the following should be kept in mind:

1. An interval will be selected which is an actual interval that was processed by the system. **This interval will be similar but may not match the borderline period exactly.**

2. If the objective is to process ninety percent of all periods incurred with satisfactory service levels, the periods near this ninetieth percentile borderline must be processed and service levels must be satisfied.

3. In the future, ninetieth percentile intervals will be processed and it will be required that the service objectives be satisfied. It is not likely that these intervals will have a "typical" workload mix, since "typical" is more like "average" mixes and the objective is to identify ninetieth percentile periods. However, it is likely that the workload mix will be similar to the critical performance periods identified using this criteria.

Workload Mix Impacts Occur in Two Different Ways

There are two types of workload mix to be concerned with:

- Inter-workload
- Intra-workload

Inter-workload mix is the mix of workload volumes between major workloads. For example, inter-workload mix would involve the number of IMS online transactions, the number of BMP data base DL/1 calls processed, and the number of production O/S batch jobs. These volumes are all at ninetieth percentiles for the the borderline interval.

Intra-workload mix is the mix of workload volumes within a major workload. For example, the mix of inquiry and update transactions in an IMS online transaction workload will vary. Resource requirements and response times for inquiry transactions will be much different than for update transactions. One approach is to classify transactions by resource requirements as described above. This reduces the impacts of intra-workload mix because all transactions in a class require the same resources. However, this may not be possible in every company. Therefore, two methods of accounting for proper intra-workload mix can be used:

- If classification of work cannot be done by resource use, intra-workload mix is indicated by three factors: service level, resource utilization, and workload volumes.

  Consider two different intervals. If the CPU utilization is similar, the workload volumes are similar, and the service levels are similar then it is likely that the workload mix is the same. For example, if two intervals have a CPU utilization of 80% and the workload volumes are similar, then IMS response times should be similar unless the mix of IMS transactions (intra-workload mix) is different. If IMS response times are similar and workload volumes are similar but CPU utilization is significantly different, there is probably a difference in intra-workload mix.

  Then if all three factors (resource utilization, workload volumes, and service levels) for an interval are similar to the ninetieth percentile borderline period, then the intra-workload mix should be similar.

- If classification of work can be done by resource use, intra-workload mix can be determined by each class.

  Two methods of using classes to determine intra-workload mix have been used. Both appear to give similar results.

  - Method 1.

    The *class* volume levels for the ninetieth percentile borderline interval are set at ninetieth percentiles. This will usually result in higher total workload volumes.

  - Method 2.

    The *total* workload volume for the ninetieth percentile borderline intervals are set at ninetieth percentile levels. The class levels are based on the average percent of the total workload volume that each class normally represents. The calculation in **TABLE III** shows how the class volumes are determined

  Method 1 sets workload *class* volumes at ninetieth percentiles while Method 2 sets *total* workload volumes at ninetieth percentiles and class volumes are calculated based on an *average* mix. Both appear to give similar results. However, Method 1 appears to be more restrictive and may not provide enough tolerance in identifying critical performance periods.

TABLE III

Class(i) volume = (ninetieth percentile workload volume) x weight(i)

where:

$$\text{weight}(i) = \frac{\sum_{\text{over all intervals}} \text{CLASS(I) VOLUME}}{\sum_{\text{over all intervals}} \text{TOTAL WORKLOAD VOLUME}}$$

i = the number of class within the workload

Figure 3. **Calculations to determine Class volumes within a workload:** The workload volumes for a ninetieth percentile borderline period are used to determine the workload volumes of each class within the workload. It is assumed that the proportion of class i volumes to total volumes for the workload will remain the same during the "critical" performance periods.

A SINGLE INDICATOR FOR EACH PERIOD SIMPLIFIES INTERVAL SELECTION

With many factors to consider in determining critical performance periods (15 for the example used in this paper), identifying representative intervals of critical performance can be complex. Developing a single indicator for each interval that reflects the similarity to the ninetieth percentile borderline interval simplifies the interval selection process.

This single indicator must be a combination (or function) of all the factors. However, combining the factors must be done carefully. Simply adding the factors together would result in the domination of the workloads by measurements of larger magnitude. For example, BMP data base DL/I volumes may exceed millions while IMS response time will likely be less than 10 seconds. This problem is similar to the problem that occurs in clustering. The problem is resolved in clustering by scaling the factors. A scaling solution will also work in this application.

All Factors Must be Scaled Before Combining into a Single Indicator Value

There are many methods of scaling. One common method is known as the zscale which scales each value by subtracting the mean and dividing by the standard deviation. The ZSCALE equation is shown as equation 1 in **TABLE IV.**

The scaled factor can be interpreted as the number of standard deviations from the mean of each factor. This scaling is successful in providing a uniform scale for all factors. However, in addition to obtaining a uniform scale for all factors, it is also important to know how close each factor is to the ninetieth percentile borderline interval.

Equation 2 in **TABLE IV** is modification of the common zscale which also indicates how close an interval is to the ninetieth percentile borderline period. This scaled factor can be interpreted as the number of standard deviations each factor is from the ninetieth percentile. The scaled factor then accomplishes two things:

- Puts all factors on a uniform scale.
- Relates how close the factor is to the desired level.

A Single Indicator Can be Found Through a Weighted Sum of Scaled Factors

By simply adding all scaled factors together for an interval, some indication of the similarity to the ninetieth percentile borderline period can be determined. If all of the factors are exactly equivalent to the ninetieth percentile values, then the sum of the scaled factors will be zero (i.e. each scaled factor would be zero). If the sum of the scaled factors is less than zero, then the interval generally is lower than the borderline interval. If the sum of the scaled factor is greater than zero then the factors are generally higher than ninetieth percentile borderline periods.

However, some factors used to calculate the interval indicator may be more important than others. For example, because much of a capacity planning process is done to ensure adequate CPU resources, it may be more important for CPU utilization to be close to the ninetieth percentile. It may be more important that the workload volumes of a workload that uses 40% to 50% of the resources be nearer the ninetieth percentile than the workload volumes of a a workload that uses only 10% of the resources.

Because there are factors which are more important in determining the similarity of an interval to the ninetieth percentile borderline period, they should be weighted

TABLE IV

1. The ZSCALE commonly used in clustering to give each factor a uniform scale

$$\text{scaled factor} = \frac{\text{factor} - \text{mean of factor}}{\text{standard deviation of factor}}$$

2. The modified ZSCALE using ninetieth percentiles instead of the mean

$$\text{scaled factor} = \frac{\text{factor} - \text{ninetieth percentile of factor}}{\text{standard deviation of factor}}$$

Figure 4. Scaling of factors: The first equation is a typical calculation to scale a variable to a common distribution with mean = 0 and variance = 1. The second equation is used in this paper to scale the factors to uniform variance. However, equation 2 is a modification of equation 1 which centers the scaled factors around the ninetieth percentile.

appropriately. Therefore, the single indicator factor can be calculated as shown in **TABLE V.**

THE INDICATOR CAN QUICKLY IDENTIFY REPRESENTATIVE PERIODS

The following criteria can be applied to the indicator value to determine the similarity of an interval to the ninetieth percentile borderline period:

1. Indicator > 0 : higher than ninetieth
2. Indicator = 0 : exact match to ninetieth
3. Indicator < 0 : lower than ninetieth

The "ideal" periods for modeling are periods where the indicator is 0. It is unlikely that any period will have an indicator of 0. Therefore, periods with indicators "close" to 0 will be good modeling periods. Experience has shown that the following guidelines should help select representative periods:

$$-.26 < \text{indicator} < .26$$

TABLE V

$$\text{indicator} = \sum_{\text{over all i}} (\text{scaled factor(i)} \times \text{weight(i)})$$

Where:

scaled factor(i) = the scaled value of factor i (as defined above)

weight(i) = the weight of factor i (based on importance)

and $\sum_{\text{over all i}} \text{weight(i)} = 1$

Figure 5. Calculation of the indicator factor for an interval: The table shows the calculation of a single indicator factor for each interval. This factor can be used to determine the similarity between a specific interval and a ninetieth percentile period. However, this equation contains offsetting factors that can effect the indicator value.

THE WEIGHTS USED IN CALCULATING THE INDICATOR FACTOR CAN BE OFFSETTING

CPU utilization and workload volume factors have a compound effect on the indicator. Consider the following:

First - Workload volumes and CPU utilization are both low.

When this occurs, the scaled factor for CPU utilization will be negative. The scaled factor for the workload volumes will also be negative. An interval that is slightly below the ninetieth percentile borderline period should have slightly lower workload volumes and slightly lower CPU utilization. When the factors are combined through the weighted sum calculation, the results are a lower negative indicator.

Second - Workload volumes are low and CPU utilization is high.

When this occurs, the scaled factor for CPU utilization will be positive and the scaled factors for the workload volumes will be negative. It is possible that when the indicator is calculated using the weighted sum, that the positive CPU factor and the negative workload volume factors will offset each other and an indicator value near 0 may result.

Similar analogies can be made for the other two possible conditions where CPU utilization is low and workload volumes are high, or CPU utilization is high and workload volumes are high. The weighted sum calculation then favors periods where CPU utilization is high and workload volumes are low, or CPU utilization is low and workload volumes are high. However, these periods are the less "representative" periods. The periods where CPU utilization is low and workload volumes are low, or the periods where CPU utilization is high and workload volumes are high are more representative (if they are close to the borderline).

The offsetting factors of CPU utilization and workload volumes can be corrected in the indicator calculations. The indicator calculation is made by subtracting the weighted scaled factor from the sum of all of the other weighted scaled factors. The final indicator equation is shown in **TABLE VI.**

ADDITIONAL CRITERIA WILL BE NECESSARY FOR THE FINAL INTERVAL SELECTION

Using the indicator value, a small number of intervals should be identified as candidates for modeling periods. The final modeling period selection can be done using the following additional criteria.

- The interval should represent the current configuration.

  The more current an interval is, the more likely that the interval will represent the current configuration. The last change to the configuration and the possible performance impacts of that change should be considered.

- Does the interval represent normal processing

  Specifically, will the processing done during an interval continue to be done in the future or was the

TABLE VI

$$\text{indicator} = \left(\sum_{\substack{\text{all factors but} \\ \text{CPU utilization}}} (\text{scaled factor(i)} \times \text{weight(i)}) \right) - (\text{scaled CPU utilization factor}) \times (\text{weight of CPU utilization})$$

Where:

scaled factor(i) = the scaled value of factor i (as defined above)

weight(i) = the weight of factor i (based on importance)

$$\text{and} \quad \sum_{\substack{\text{over} \\ \text{all i}}} \text{weight(i)} = 1$$

Figure 6. Modified indicator factor calculation: The table shows the modified calculation of a single indicator factor for each interval. The modification adds all factors except CPU utilization which is subtracted. This moves the value of the indicator farther from 0 (the ninetieth percentile borderline value) when workload volumes are high but CPU utilization is low or when workload volumes are low but CPU utilization is high.

processing a one-time function such as the initial population of a new data base etc.

- Restricting the acceptable CPU utilization range.

  The indicator value may allow a wider range of CPU utilization than may be desired. Adjusting the CPU utilization weight factor will help. However, if the CPU utilization weight factor is too high, the selection process will tend to ignore the other factors and some recommended periods may not be representative. An alternative is to keep the CPU weighting factor in the indicator calculation lower and then eliminate some recommended periods by establishing an acceptable range of CPU utilization (which should be around ninetieth percentile CPU utilization).

THE INTERVAL INDICATOR CAN SAVE TIME AND IMPROVE PLANNING.

The indicator method of interval selection can save time and improve the planning process because:

1. Periods are not modeled and later determined to be unrepresentative.
2. The cyclical nature of the processing is automatically handled.

   It is still important to understand the processing cycles and efforts should be made to do this. However, the indicator method assures that critical periods in the cyclical patterns are modeled.
3. The number of representative intervals are objectively reduced to a reasonable number from which an analyst can quickly select one.
4. It is assured that the periods which are modeled are representative.

This technique will result in a faster decision on selecting an interval for modeling and a better interval selection.

THE INTERVAL INDICATOR CAN BE USED IN ALTERNATIVE WAYS.

For model interval selection, there are two methods of implementation depending on the planning and modeling frequency. One method is to select intervals on a quarterly basis. is then selected from the previous quarter. The ninetieth percentiles and standard deviations used in calculating the scaled factors (which are eventually used in developing the indicators for each interval) are calculated quarterly at the time the model interval is selected.

Another method of developing indicators is to update the ninetieth percentile and standard deviation values used in the scaled factors on a regular basis. Then an indicator value for any recent period can be developed based on the most recent ninetieth percentiles and standard deviations. This approach has two advantages:

1. Identification of representative periods can be readily done for any current period.

   Specifically, when selecting a period from the previous day to be analyzed, the indicator can quickly determine what periods might be most representative.
2. If a performance problem occurs in a current period, it can be determined if this is a period above the ninetieth percentile borderline interval.

   If the period where the performance problem occurred is a period for which service levels are not expected to be satisfied, it may not be the necessary to analyze the performance problem. Some discretion is necessary in a decision such as this, but the indicator value can help the decision process.

THERE ARE SIX ADVANTAGES OF THIS INTERVAL SELECTION METHODOLOGY.

The interval selection methodology presented in this paper has many benefits. Six advantages of using this methodology are:

1. Intervals selected for modeling are representative.
2. The interval selection process becomes more objective.
3. The planning process will be improved by more clearly defining current performance/capacity levels.
4. The interval selection process is based on service objectives.
5. The process can eliminate wasted time modeling unrepresentative periods.
6. The indicator values can be used to reduce performance analysis of periods for which service levels would not be expected to be satisfied.

REFERENCES

1. MVS Product Users Guide, BGS Systems, 1987 (Document No. 86077-00)
2. MICS Capacity Planning Guide, Marino and Associates, 1987
3. MICS MVS Model Generator Component, Marino and Associates, 1986

TOOLS AND METHODS AVAILABLE FOR IN-DEPTH VM PERFORMANCE ANALYSIS

Deborah C. Christopher
Duke Power Company

ABSTRACT

VM performance analysis and tuning can be a very complicated process. The normal performance analysis tools (VMMAP, SMART) lack key data that is required to adequately characterize a VM system. However, additional information can be gathered using simple user-written programs and methods, and using other software packages. Topics to be discussed include 3880-23, 3880-21, PROFS, 3090-200 migration, the latest releases of VM/HPO, the VM real storage constraint, and some ways to analyze their effects on VM performance.

1. INTRODUCTION

The key to successful VM performance analysis is having multiple sources of performance data. No one source should be depended on to thoroughly represent a VM system. This statement is especially true for a large HPO system. At Duke Power, we currently execute on a 3090-200E with 64 Mb of real storage, 128 Mb of expanded storage, two 3880-E23 cache controllers with one 3380-AA4 each (four single density 3380's), six full strings of single density 3380's (sixteen volumes each), and one full string of dual density 3380's (sixteen volumes each). At this writing, there are 1025 logged on users; this number is expected to grow 45 per month through 1987, and approximately 30 per month in 1988.

Instead of giving an overall description of each source of performance data, a specific use of the data will be presented as it relates to solving a particular problem or gaining insight into a situation or trend.

2. REAL STORAGE CONSTRAINT

2.1. BACKGROUND AND DESCRIPTION

The real storage constraint condition that exists for installations with large numbers of users logged on (1000+) may hinder the efficient use of IBM 3090 processors in the current VM environment. Since VM/SP HPO Release 1 was available in early 1982, there have been advances in processing power, DASD, and operating system software. Concurrent with these technological advances is the emergence of large VM communities on a single machine such as Duke Power's PROFS workload on a 3090-200E. However, since VM/SP HPO was designed with 24-bit addressing (16 Mb of storage is addressable), CP is limited to executing below the 16 Mb line. CP must reference (< 16 Mb) a V=R region, dynamic paging area (DPA), free storage (and prime free storage), and the trace table. Refer to Figure 1 for a map of real storage as it currently exists on Duke Power's 64 Mb 3090-200E.

Major Area	Size	Minor Area	Size	Address Range
Hardware	448K	Control Storage	448K	03F90000-03FFFFFF
Extended Storage	48,704K	Extended Dpa	48,704K	01000000-03F8FFFF
CP Fixed Storage	7,432K	Prefixa	4K	00FFF000-00FFFFFF
		Prefixb	4K	00FFE000-00FFEFFF
		Prime	240K	00FC2000-00FFDFFF
		High Free	6,928K	008FE000-00FC1FFF
		Trace Table	256K	008BE000-008FDFFF
Dynamic Area	8,124K	Dynamic Paging	8,124K	000CF000-008BDFFF
CP Nucleus	824K	Low Free	3,560	000CE218-000CEFFF
		CP Nucleus	820K	00001000-000CE217
PSA	4K		4K	00000000-00000FFF
Total	64M			

Figure 1.

As early as Summer 1985, it became apparent that Duke Power would have to slow down expansion of PROFS because of the real storage constraint below the 16 Mb line (we did not call it that at the time). Free storage allocation was increasing with the rising number of users logged on. Ten megabytes seemed to be about the most free storage that could be allocated and still have a useful Dynamic Paging Area. Table 1 shows the relationship between free storage allocation and the number of logged on users, as was experienced at Duke Power Company.

Table 1

APRIL 1985 - NOVEMBER 1986

Free Storage Alloc.	Users Logged On
4 Mb	320
5 Mb	400
6 Mb	500
6.5 Mb	600
7 Mb	700
8 Mb	750
8.5 Mb	800
9 Mb	850

Free storage is used for CP control blocks - page and swap tables, and control blocks for spool files, virtual channels, virtual control units, and virtual devices. The amount of storage set aside as free storage is determined by the number of pages coded in the FREE operand of the SYSCOR macro in DMKSYS. The default is three pages for the first 256 Kb of storage, one page for each 64 Kb after that, additional pages are allocated if the system used extended storage, one page is allocated for every 256 Kb of > 16 Mb storage, and the default is 25% higher in AP and MP modes.

Prime free storage is an area within free storage dedicated to cache-aligned (16 doubleword) control blocks. The default allocation for prime free is 10% of the free storage allocation. For example, if free storage is set at 6 Mb (1536 pages), then the default allocation for prime free would be 154 pages. Sixty (60) pages have been enough for us; prime free storage utilization averages 55% for 15-minute summarizations. The prime free setting is coded in the PRIME operand of the SYSCOR macro in DMKSYS. Planning guides usually give a recommendation on the size of FREE and PRIME. However, we have needed more free storage and less prime free than IBM has estimated (approximately 2.5 pages per user HPO 3.4 and 1.4 pages per user under HPO 4.2). Once the system is running, the settings should be reevaluated using information from the Virtual Machine Monitor Analysis Program (VMMAP) OUTFREE1 report (for free storage), and OUTFREE2 report (for prime free storage).

OMEGAMON/VM (Candle Corporation) can provide detailed real-time information about free storage utilization:

1. TTFR - summary of free storage utilization by userid

2. FREA - number of CP free storage frames available

3. FREE - number of CP free storage frames defined in DMKSYS

4. FREL - number of CP free storage frames in use

5. FREX - number of CP extended free storage frames

VM Real Time Monitor (SMART) provides a summary of free storage calls in the FREE display. SMART's LOGMSG 6 will indicate when CP FREE STORAGE is extended beyond the number of pages specified in the LIMIT.

2.2. MINIMIZING THE REAL STORAGE CONSTRAINT

IBM has made several recommendations on reducing low storage requirements in general, and on reducing free storage requirements in particular [3]. These recommendations include the following:

1. Migrate to HPO 4.2.

2. Reduce the size of the CP nucleus by tailoring the standard load lists.

3. Reduce the size of the CP internal trace table, if feasible. Settings other than the default are specified in the SYSCOR macro of DMKSYS, TRACE option.

4. Reduce the number of monitor buffers. Watch for VM monitor suspension records. This parameter, BUFFS, is also set in the SYSCOR macro of DMKSYS.

5. Reduce SET RESERVE (if any) as much as possible.

6. Check free storage and prime free for overallocation and adjust.

7. Encourage inactive users to LOGOFF, thereby freeing up their free storage pages.

8. Eliminate unnecessary devices from directory entries; each virtual device control block requires free storage.

9. Number virtual device addresses consecutively to minimize the number of virtual channel and control unit blocks.

2.3. FREE STORAGE GAINS WITH HPO 4.2

With approximately 850 users logged on when Duke Power implemented HPO 4.2, free storage requirements dropped dramatically - from 9 Mb down to 6 or 7 Mb. This drop can probably be attributed to the automatic migration of the page and swap tables of inactive users. HPO 5.0 should reduce free storage requirements even more.

3. 3880-23 CACHE CONTROLLERS

3.1. EVALUATING CACHE CANDIDATES

Planning is the key to making effective use of IBM 3880-23 cache controllers; the Model 23 is used to cache non-paging data. When cache is properly used, significant reductions in response time can be realized, especially in a PROFS environment. Good candidates for caching include data with relatively high activity and high read to write ratio. Other questions to consider are:

- Will better response times on this I/O measurably reduce end-user response time?

- How much cache is available?

- How many minidisks on the volume should be cached?

To determine minidisk access rates and read/write ratios, the SEEKS class of VM Monitor data should be enabled. However, enabling SEEKS for an entire eight or nine hour period would produce an unmanageable amount of data. A better alternative is to randomly enable the SEEKS monitor class, divide the prime time period into five-minute intervals and randomly choose a percentage of those intervals (10% works well). This random sampling method should provide the kind of information necessary to characterize minidisks for caching purposes. VMMAP can produce four reports from SEEKS data:

1. Detail by VOLSER, starting cylinder, inactivity included.

2. Summary by VOLSER.

3. Summary by VOLSER, sorted by descending usage.

4. Detail by extent, sorted by descending usage, inactivity excluded.

Each SEEKS detail report lists volser, start and end cylinder, owner id, virtual address, total seeks, percentages of the device and system that the seeks constitute, and a read percentage. The read percentage is only available on systems running VM/SP HPO 4.0 and up. This value is used to determine the read/write ratio for the data. That data, combined with the size the minidisk and percentage used should be enough information to select preliminary caching possibilities. The percentage of the minidisk that is currently being used can be extracted (using disk queries) and multiplied by the size of the minidisk to determine the actual number of blocks/cylinders of data. All of the information can be combined (using SAS, for example) to produce a list of frequently used minidisks and their access rates, read percentage, and DASD occupied. Certain data areas should be excluded from the list: PAGE and SWAP areas, SPOOL, and TDISK areas. The effect of caching data on end-user response time can be estimated by modeling or tested through trial and error.

Data can be cached by volume or by minidisk. Duke Power had E23 cache controllers prior to running HPO 4.2; at that time, we used VM/SP CMS Support for 3880 Model 13 and 3880 Model 23. The Program Offering allowed us to designate which CMS minidisks should use the cache, and also provided a new VM Monitor record (MN606 - VM Monitor Cache Performance Data). This monitor record included basic information about I/O requests, and read and write hits; we created our own Cache Activity reports using these records. With HPO 4.2, VM Monitor now records cache performance data and VMMAP provides a basic analysis of the data.

3.2. ANALYZING CACHE PERFORMANCE

Using this process at Duke Power, for two 3880-E23's (16M each = 23.56 cylinders (3380 DASD)) the following minidisks were placed at the top of the list:

SYSADMIN 399 - 17 cyl (3380 DASD) * 61% used = 10.37 cyl = 7.04 Mb

SYSADMIN 498 - 5 cyl (3380 DASD) * 96% used = 4.80 cyl = 3.26 Mb

MAINT 190 - 31 cyl (3380 DASD) * 84% used = 26.04 cyl = 17.69 Mb

Caching this data reduced the response time from 22 to 3 MSEC/IO in each case, with a read/hit ratio of approximately 100%. This reduction brought about a noticeable improvement in the elapsed time for logons to our PROFS system, and is a significant factor in our ability to support more than one thousand logged on users today.

The VM Real Time Monitor (SMART) provides some information on read/hit ratios in the CACHDBSE display. Figure 2 is an example of the 3880-23 display. Devices 0300 and 0800 contain cached data. Notice that there is one line on the display for each storage director. VMMAP's 3880-23 Subsystem Activity Log also summarizes cache usage including cache hit ratios, I/O rates, and read percentages.

VM/370 CPU3090 SERIAL 71982 65088K DATE 08/20/87 START 08:00:50 END 15:49:53

A V E R A G E S

DIRECTOR AND ADDRESS	TOTAL I/O RATE	OVALL READ H/R	<----NORMAL----> I/O RATE	READ _H/R	READ _PCT	<--SEQUENTIAL--> I/O RATE	READ _H/R	READ _PCT	ICL RATE	BY-PASS RATE
C0 0300	11	100	11	100	100	0	0	0	0	0
C3 0300	11	100	11	100	100	0	0	0	0	0
C0 0301	0	0	0	0	0	0	0	0	0	0
C3 0301	0	0	0	0	0	0	0	0	0	0
C0 0302	0	0	0	0	0	0	0	0	0	0
C3 0302	0	0	0	0	0	0	0	0	0	0
C0 0303	0	0	0	0	0	0	0	0	0	0
C3 0303	0	0	0	0	0	0	0	0	0	0
C2 0800	0	99	0	99	100	0	0	0	0	0
C1 0800	0	99	0	99	100	0	0	0	0	0
C2 0801	0	0	0	0	0	0	0	0	0	0
C1 0801	0	0	0	0	0	0	0	0	0	0
C2 0802	0	0	0	0	0	0	0	0	0	0
C1 0802	0	0	0	0	0	0	0	0	0	0
C2 0803	0	0	0	0	0	0	0	0	0	0
C1 0803	0	0	0	0	0	0	0	0	0	0

Figure 2.

4. 3880-21 CACHE CONTROLLERS

The IBM 3880-21 cache controller was designed for use by the paging subsystem. At Duke Power, before our HPO 4.2 system was moved from a 3081-K to a 3090-200E, two 3880-J21 paging cache controllers were used. The utilization of the J21's, along with the associated tuning that was performed resulted in a dramatic improvement in host response times:

Trivial Response - 45% improvement

Average Response - 21% improvement

The J21's were configured symmetrically, with the ALTCU at the same address as the primary storage director. Using this method, the second storage director is also a second channel path. If ALTCHAN had been used, the second control unit would not have been recognized, causing unnecessary control unit busy conditions.

For each real paging address defined to the 3880-21, three other paging addresses are generated. For a real address 200, for example, the generated paging addresses would be 204, 208, and 20C. For a real paging address 201, the generated paging addresses would be 205, 209, and 20D. After a 3880-21 actuator has been selected to service a paging I/O, the first paging address is looked at. An I/O is issued to that address if it is free. If it has an active I/O or if it is already scheduled, the second paging address is looked at, then the third, etc. If all addresses are scheduled or active, the I/O will be queued to the address with the smallest queue. The I/O is distributed over the paging addresses as described in Table 2.

Table 2

Distribution of I/O

Paging Address	Percent of I/O	MSEC/IO
1st	68	4.1
2nd	18	9.2
3rd	14	12.1
4th	< 1	3.6
	Average Page Service Time	6.138

5. EXPANDED STORAGE

At present, there seems to be no accurate way to calculate response time per page or swap from expanded storage. Fortunately, this lack of information should not significantly impact the duties of a performance analyst. The response time for expanded storage operations should be positively reflected in lower system response times. Data is available on the allocation and usage rates of expanded storage.

The SMART PAGEAREA display includes allocation and deallocation rates, reads and writes, start I/Os per second, and percent full of each SW, PP, PG, PM, and PS area. Refer to Figure 3 for an example. VMMAP includes similar information on allocation rates, reads, and writes.

VM/370 CPU3090 SERIAL 71982 65088K DATE 08/24/87 START 08:00:17 END 08:49:39

A V E R A G E S

VOLSER	TYPE	ADDR	LO CYL	HI CYL	ALOC /SEC	DEAL /SEC	RDS /SEC	WRTS /SEC	SIO /SEC	AVG. CYLS	% FULL	AVG. PAGES
******	SWAP	*PS*	1	8	4	4	40	41	82	8	99	8181
VMCS06	SWAP	0C00	1	884	0	0	7	7	1	884	1	2606
VMCS33	SWAP	0F04	1	884	0	0	7	7	1	884	1	2771
VMCS27	SWAP	0D00	1	884	0	0	7	7	1	884	1	2749
VMCS24	SWAP	0B00	1	884	0	0	7	7	1	884	1	2758
VMCS23	SWAP	0A00	1	884	0	0	7	7	1	884	1	2658
VMCS14	SWAP	0F00	1	884	0	0	7	7	1	884	1	2788
VMCS10	SWAP	0E00	1	884	0	0	7	7	1	884	1	2678
VMCS07	SWAP	0703	1	884	0	0	7	7	1	884	1	2845
******	PPAG	*PS*	9	32	19	16	15	19	35	24	93	23160
VMCS02	PAGE	0E02	1	884	1	0	0	1	1	884	0	1336
VMCS03	PAGE	0F02	1	884	1	0	0	1	1	884	0	1359
VMCS21	PAGM	0303	1	884	0	0	0	0	0	0	0	0
VMCS22	PAGM	0B03	1	884	0	0	0	0	0	0	0	0
VMCS04	PGPS	0D0E	1	884	0	7	7	1	9	283	17	23228
VMCS13	PGPS	0B0E	1	884	0	8	6	1	8	282	17	23165

6. ANALYZING PROFS WORKLOADS

6.1. VM COMMAND MONITORING

In the current VM environment, the choices for performance monitors are limited, especially in the area of command monitoring. Various mechanisms are available for MVS with which to capture detailed information by userid, job, TSO command, CICS transaction, etc. The lack of information of this type for VM does not suppress the need for it. Command level information can be a valuable tool in VM performance analysis, tuning, and capacity planning.

VM Monitor can provide some data, but it is not very comprehensive. The USER (Class 4) records from VM Monitor show resource usage statistics for the interval (usually 60 seconds) for each user in the system. RESPONSE (Class 1) records show some terminal activity information, but the input and output terminal data collected is limited to first 128 bytes [6].

The VM Accounting data also includes user-specific information, but in less detail than the VM Monitor data. The virtual machine user records are stored when a user ends his terminal session or when the ACNT command is issued. The resources used (connect time, CPU time, page reads and writes, sios, spool information, etc.) are summarized for the session. Obviously, accounting data is not retained at the command level either.

6.2. PROFS COMMAND MONITORING

There are few command monitoring facilities available. IBM provides exits with PROFS that allow data collection with customized exit routines [4]. These exits, called by PROFS, are sent information on the size and frequency of messages, notes, and documents sent and received.

The exits are listed below along with a brief description of when each is called.

- EPRACCTG - called at the start and end of PROFS sessions, PROFS commands, non-PROFS commands from the PROFS command line, when a message or note is sent, when a message is received, and when a document is stored.
- EPRRACCT - called whenever a user opens the mail.
- EPRMACCT - called whenever mail is sent or received.
- EPRLACCT - called whenever an item is stored in or retrieved from the document or calendar data bases.

Each exit can collect several different observations, and there are some restrictions and guidelines that must be considered when coding each exit. Additionally, since all PROFS users would be executing these exits for potentially each task performed, the amount of data collected could be prohibitive. The *PROFS Programmer's Guide* cautions that "If you enable the accounting user exits, you risk degrading performance." Such notes regarding system performance are rarely present, but when they are they should be adhered to and certainly not taken lightly. All of these cautions should be magnified for large PROFS communities on a single machine. The documentation also suggests avoiding spooling by using SMSGs to a PROP (Programmed Operator) machine.

6.3. OTHER OPTIONS

There are a handful of performance monitors that collect information in varying degrees of detail at the command level. Duke Power has successfully used CMAP (Command Monitor/Analysis Package from VM/CMS Unlimited, Inc.) for studies requiring command information. One study involved determining the number of PROFS ids that use the REMINDER function of PROFS. Using SET CLOCK OFF as the default for all PROFS users was under consideration. Eliminating activity caused by timer "pops" would reduce the resource consumption by truly inactive users. Setting the clock off would also reduce the effectiveness of reminders since the reminder would not be displayed during periods of inactivity without a timer pop. However, if the REMINDER function is used by a small percentage of users, it may not be an issue.

Certain minor changes in the PROFS environment are necessary to take advantage of CMAP monitoring. Once the changes were made and after gaining more experience in using the product, we collected data that showed that 7% of our PROFS users had used the reminder functions.

7. HOME-GROWN PERFORMANCE DATA BASE

Instead of running the VM Monitor data through an expensive data base system (expensive to buy, maintain, support, use, and store), in-house software was developed to retain key parts of VMMAP and SMART in a SAS data base for use in daily performance analysis and short-term capacity planning and trend analysis. The data base provides added flexibility in data manipulation and trend analysis beyond VMMAP ACUM files. Variables in the data base come from the following VMMAP reports:

1. Monitor Statistical Summary (OUTSTAT)
2. User Response Time Analysis (OUTRESP)
3. User Resource Utilization Summary (OUTUSER)
4. Disk and Tape I/O Activity Summary (OUTDASD)
5. Channel Activity Summary (OUTCHAN)

In many cases, data from this performance data base is enough for analysis. However, if more detailed information is needed, there are several ways to derive it:

1. Produce a more detailed VMMAP report. For example, if you need to know system resource utilization (CPU busy, paging, swapping, storage) by minute, then run VMMAP with an interval of one (1). This interval is specified in the MON PARM file; this file is used to select VMMAP reports to be produced.
2. Process the VM Monitor data.
   a. The format of the VM Monitor tape is described in the *Systems Programmer's Guide*. For more information on individual fields, consult *VM/SP HPO Data Areas and Control Block Logic - CP*.
   b. The time-of-day field must be decoded.
      1) Each record has a header portion. BYTE 1 is the Monitor Class, BYTE 2 is the Monitor Code, and BYTES 3-7 are the time-of-day. The Systems Programmer's Guide notes, "Time of day occupies 2 fullwords in storage, with the rightmost 12 bits zeros. The rightmost 2 bytes and the leftmost byte are ignored, giving 16-microsecond accuracy instead of 1-microsecond." More simply stated, use the SAS informat PIB5.6 (or the equivalent), positive integer binary 5 bytes long, 6 decimal places.
      2) The time-of-day contained on each data record is relative to the Date and Time of data collection on the CLASS 0 Code 97 record.
3. Process the VM Accounting data.
   a. The VM Accounting dataset is much smaller than the VM Monitor dataset. The timestamp needs no conversion, and the records are fixed length, 80 bytes.
   b. This data can provide quick information on the number of logons and logoffs per day. Using user accounting fields, the amount of resources consumed by groups, divisions, and departments (for example) can be reported. In general, the basics about each terminal session can be extracted.
4. Process the operator's console log.
   a. The operator's console log is the only place that terminal id can easily be determined. The terminal id can be used to analyze the distribution of ids across lines.
   b. Exception messages from SMART can be seen in relation to other activities in the system: logons, logoffs, disconnects, reconnects, VTAM activity, and any user-generated messages. Information such as system time, spool utilization, and query users (if issued by the operator) will also be included.

Figures 4, 5, and 6 were created using data from the performance data base and SAS/Graph. Please note that a 3090-200E replaced a 3081-K in WEEK 24 of 1987.

8. SUMMARY

Understanding the performance characteristics of large VM systems can be a complex process. However, the task will be made easier and more reliable by using multiple data sources and tools. Conventional methods will not always point out potential problems, especially in the following areas:

- Real storage
- Cache controllers
- PROFS workloads

- Large VM systems

The sources of VM performance data mentioned include the following:

- VM Monitor data
- VM Accounting data
- Operator's console log
- Command information
- Real-time monitors (SMART, OMEGAMON/VM, for example)
- VMMAP

With VM/XA on the horizon, the sources of data and tools available to deal with that data will probably change. However, the basic idea behind VM performance analysis should always remain the same: the more you know, from as many sources as you can find, the better.

REFERENCES

[1] B. D. Robinson, *The VM/SP High Performance Option Performance Tuning Guide*, IBM Washington Systems Center Technical Bulletin, GG22-9392-01, November 1985.

[2] Candle Corporation, *OMEGAMON/VM Version 300 Release Guide VM/SP, VM/SP HPO, and VM/XA*, OV60-1504-20, December 1986.

[3] D. Hoag, *Managing VM/SP HPO Real Storage Constraint Below the 16 Megabyte Line*, IBM Washington Systems Center Technical Bulletin, GG66-0261-00, October 1986.

[4] IBM, *Professional Office System Programmer's Guide*, SH20-6081-1, June 1986.

[5] IBM, *VM/SP CMS Support for 3880 Model 13 and 3880 Model 23 Program Description/Operations Manual*, Program Number: 5798-DRJ, SH20-6537-0, November 1984.

[6] IBM, *VM/SP HPO Systems Programmer's Guide*, SC19-6224-6, December 1985.

[7] A. H. Shub, "PROFS Capacity Planning", Share 68, March 1987.

[8] VM/CMS Unlimited, Inc., *CMS Command Monitor/Analysis Package, User's Guide and Installation Guide*, Second Edition, April 1986.

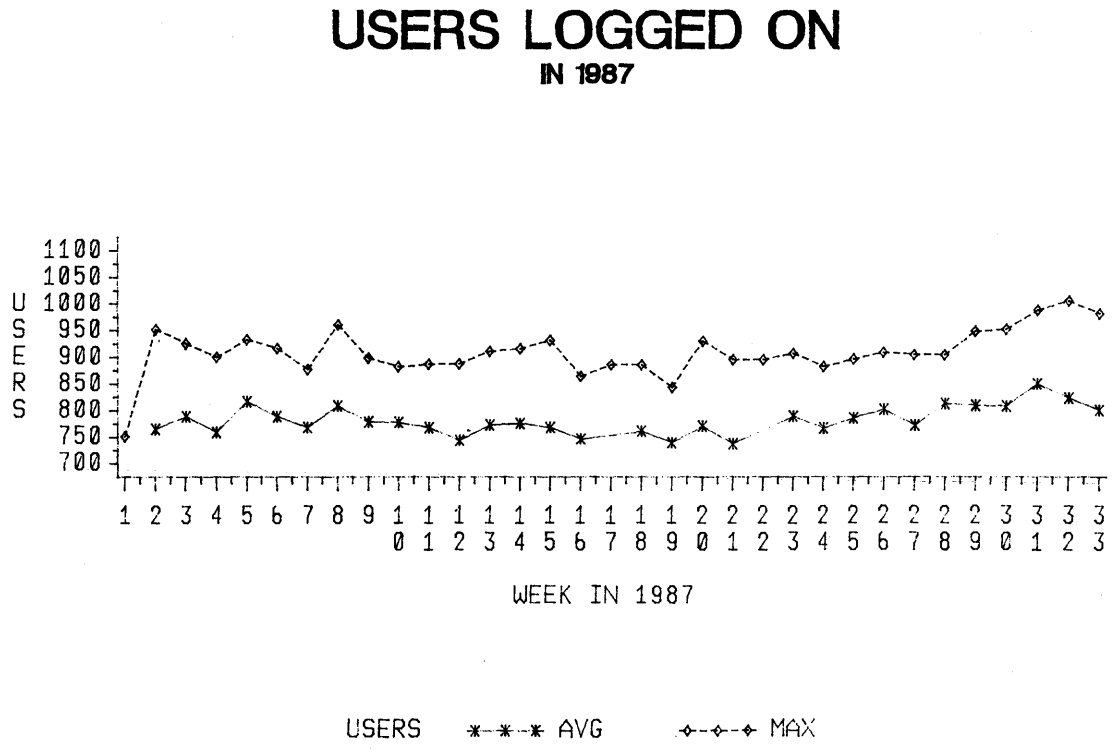
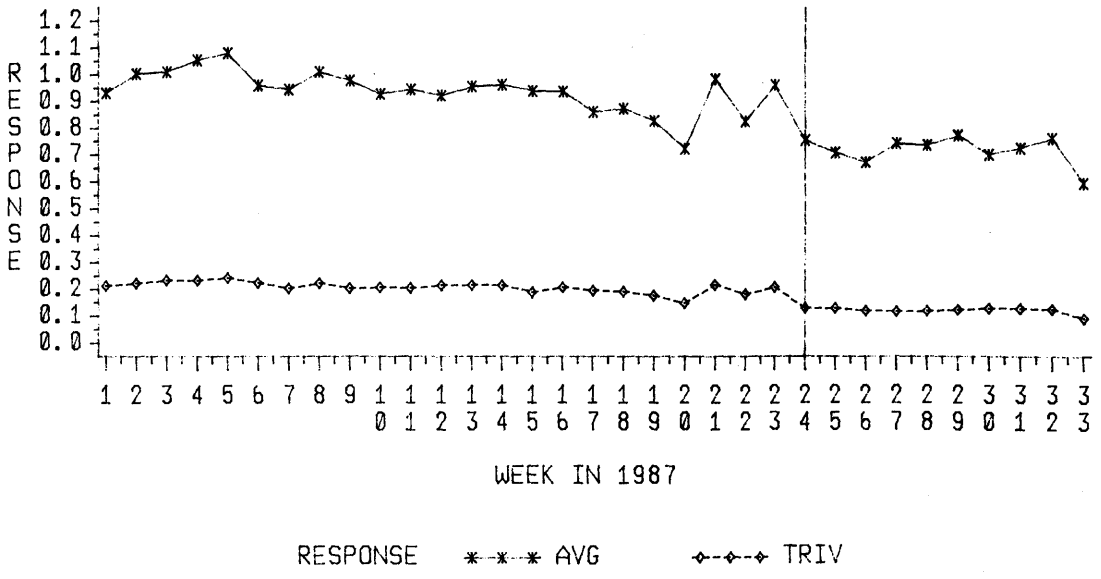


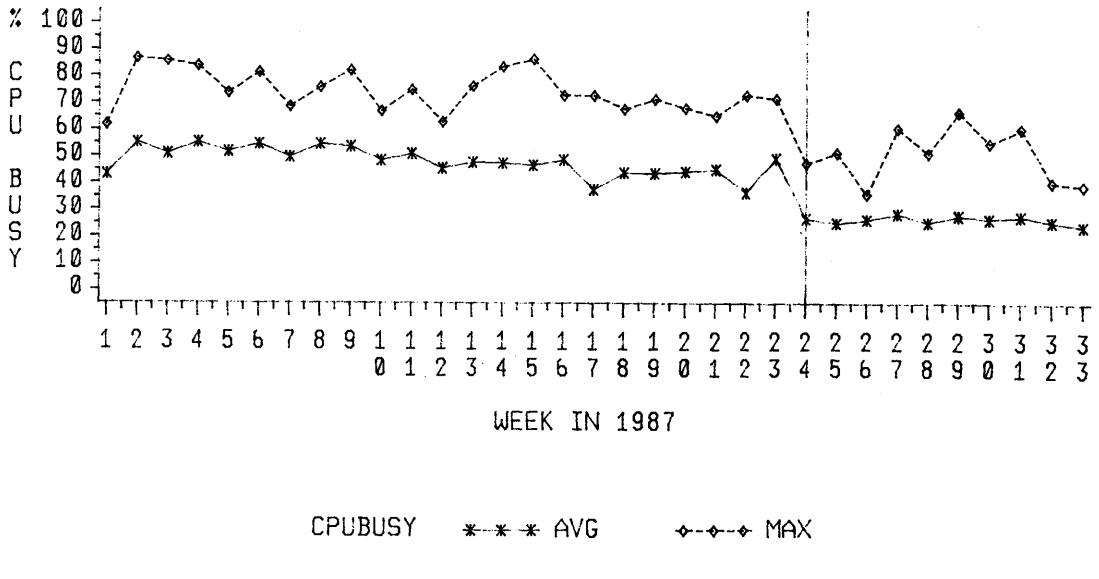
FIGURE 4.

TRIVIAL AND AVERAGE RESPONSE TIMES
IN 1987



WEEKS 1-23 ON 3081-K
WEEKS 24-33 ON 3090-200E
FIGURE 5.

CPU UTILIZATION (MAX AND AVG)
IN 1987



WEEKS 1-23 ON 3081-K
WEEKS 24-33 ON 3090-200E
FIGURE 6.

CICS AND DB2: A MARRIAGE MADE IN HEAVEN?

Jason Shane
Coldwell Banker Residential Group

James Cook
Data Strategies, Inc.

ABSTRACT

The use of IBM's CICS and their DB2 data base systems continues to grow in popularity, yet most are unaware that a basic inconsistency between CICS and DB2 architectures exists which can cause response time degradation, user dissatisfaction with the application and additional network traffic.

This paper documents in detail each of the known problems and discusses their impact on user productivity and system resources. In addition, the CICS architectural features, DB2 restrictions, and SNA protocols involved are reviewed and their effects on the application design are discussed.

Because architectural incompatibilities are difficult to detect and impossible for the user community to resolve, the authors will also discuss how these problems were detected and the mechanism through which a resolution was pursued with IBM.

INTRODUCTION

Early in 1986 IBM began to stress its commitment to its relational database DB2. Significant performance enhancements were made and more were rumored. Additional incentives were generated for seriously considering the development of DB2 applications when IBM announced that its coming data repository would make DB2 an MVS prerequisite. In the midst of this confusion and uncertainty, IBM attempted to pacify a customer base concerned over the viability of its investment in existing products and strategies. To this end, Merry Quakenbush, Director of Industrial Sector Marketing for IBM in Atlanta said:

> "IBM will continue to pursue a dual database strategy for the balance of this decade. We are going to protect the investment of our customers. By 1990 however, things could change. Over time you will see IBM developing into a relational database provider."[1]

Statements of this type encouraged progressive installations to "jump on the DB2 bandwagon". Those developing online applications utilized CICS for communications by default, since no other vehicle was available. Given the lack of choices, they felt confident that IBM had clearly "marked" this approach as a migration path.

Now, with IBM's promise two-and-one-half years old, some of these "pioneers" have fully-functional, production DB2 systems. Only recently have they discovered that there were a few "unpaved stretches" on the migration path. CICS' support of DB2 is substantially different than its other support. As a result, IBM has exacted a high toll in performance. This paper endeavors to document the nature and consequences of one known problem.

THE PROBLEM AND ITS IMPLICATIONS

A large number of CICS applications have been designed to execute as pseudo-conversational transactions. To pass control from one application program to another the CICS transfer control function (XCTL) or LINK are most often used. Doing so, causes the original CICS transaction to remain in control. Several different application programs may have been executed, but only one CICS transaction was involved.

However, when using DB2 with CICS, neither the XCTL or LINK have any effect on the DB2 thread! Therefore, the XCTL and LINK will not work the same in a CICS/DB2 environment as they did in a CICS/VSAM environment. This is because each CICS application program is associated with a single DB2 plan in the CICS Resource Control Table or RCT--one application program to one plan. In other words, as a result of an architectural incompatibility between CICS and DB2, two of the most widely used of CICS functions, the XCTL and LINK will not work as before.

As a result, to execute an application program other than the one that was originally started, the first CICS transaction must terminate and then start a new CICS transaction whose definition in the RCT points to a different DB2 plan.

When the first CICS transaction ends, CICS architecture dictates that an SNA End Bracket (EB) be sent to the user's 3270 terminal. Then, before the second CICS transaction can begin, the terminal must be reacquired using the SNA BID request. Only after an SNA positive response is received, can the second CICS transaction execute.

Consider the following scenario (Follow along with the illustration in Figure 3). The user starts the menu transaction by entering "MENU." An SNA Change Direction (CD) and Begin Bracket (BB) are sent locking the user's keyboard. The MENU CICS transaction is started, obtains service from DB2 and transmits the menu screen to the user, ending the transmission with an SNA End Bracket (EB) which unlocks the user's terminal. Options 1 thru 6 are now displayed on the user's terminal.

The user now selects Option #1 and hits Enter. Another SNA Change Direction and Begin Bracket are sent locking the user's keyboard. The MENU transaction starts a second time to interpret which option the user selected. Having deter-

mined that Option #1 was selected, the MENU program now needs to pass control to the program that will execute Option #1. The CICS XCTL function cannot be used because the Option #1 program will execute a different DB2 plan than the MENU program. Therefore, the MENU transaction issues a start transaction for the Option #1 transaction (Transaction #3 in Figure 3). The MENU transaction ends by transmitting an SNA End Bracket which unlocks the user's terminal.

Before the Option #1 transaction can start, CICS must re-establish communications with the user's terminal. Since, according to convention, CICS does not know if the terminal is still available (even though CICS released the terminal only a fraction of a second ago), CICS must transmit an SNA BID to request the terminals attention. The Option #1 transaction must now wait until an SNA positive response is received, indicating the terminal is available for communications.

When the positive response is received, the Option #1 transaction starts executing, obtains DB2 services, and transmits the data starting with an SNA Begin Bracket that locks the user's terminal again. When all data has been sent, CICS sends an SNA End Bracket to again unlock the user's keyboard. The user's transaction is now complete.

Thus, to complete a single user transaction, two or more CICS transactions must be executed. This one-to-many mapping of transactions has significant impact on service levels, user satisfaction, network capacity and network management.

USER SATISFACTION OF ONLINE SERVICE

In the example, it was noted that when the MENU transaction ended, an SNA End-Bracket was transmitted to the user's terminal causing the user's keyboard to be unlocked. The user's keyboard remains unlocked for the duration of the following transaction (point 4 to point 8 in Figure 3) which could be several seconds.

When a user sees the "clock" symbol go away and no data appear on the screen, it seems to be human nature to try the last transaction again. Who knows maybe it will work better this time. However, an annoying situation occurs when the user enters a second transaction from the same terminal from which one is already in progress. The CICS node error program (NEP) generates an ATNI abend and the user's SNA session with CICS is immediately terminated. The only way to recover is for the user to call the company's help desk staff and for them to provide the necessary support to restore the user's CICS session.

To summarize, a user enters a transaction and waits. The system responds by unlocking the keyboard in the manner the user has observed many times before; only this time, no response to the user's request is received. The frustration begins. Thinking things have failed, the user tries the transaction again only to be knocked completely off the air. The user's temperature soars as he calls the company's help desk. Any help desk problems encountered about now, make the situation intolerable in the user's eyes. Information Services has failed again!

These problems occurred as a result of the first, unnecessary End Bracket that unlocked the user's keyboard. The EB was sent because the CICS XCTL was not available in a CICS/DB2 environment. As a result, multiple CICS transactions were needed; but whenever a CICS transaction ends, by convention, the End Bracket is sent.

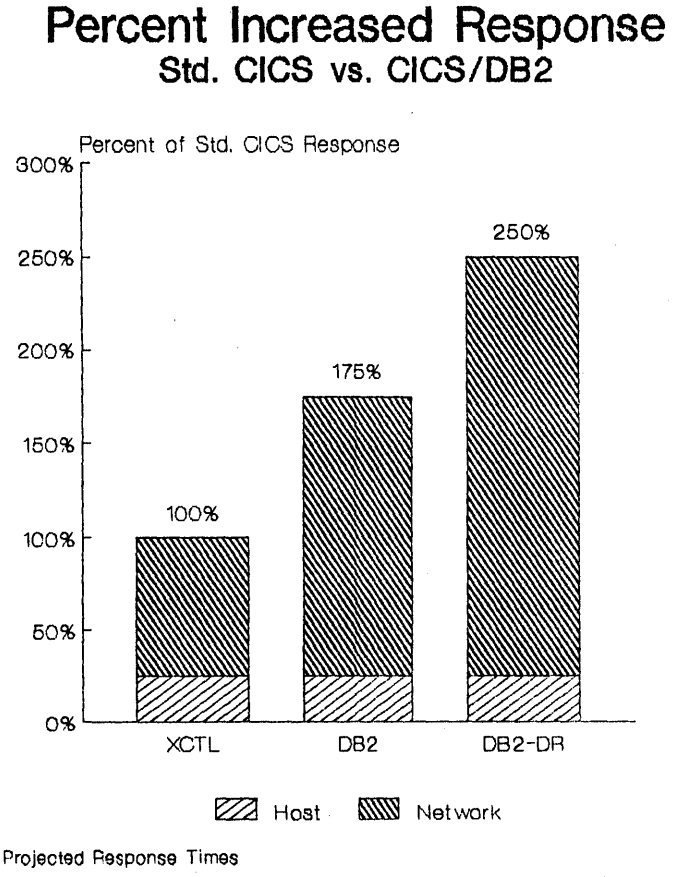
RESPONSE TIME IMPACT

In a CICS/VSAM environment, the transfer of control from one program to another (XCTL) within a CICS transaction usually occurs somewhere in the millisecond time frame--a small effect on user response time. However, the multiple transactions required in the CICS/DB2 environment have considerably more impact.

In the example, when the first CICS transaction ended, it released the terminal (point 4 in Figure 3). Before the second CICS transaction could process a complete round trip of the network was required to reacquire the terminal (refer to points 5 to 7 in Figure 3).

For locally attached users, the extra round trip of the network to BID for the terminal is typically quite fast and is generally unnoticeable to the end user. Remote users in single-domain networks, generally see a definite, perceptible, but not intolerable addition to their response time. However, when the "extra" BID must traverse the backbone of a complex, nationwide, multi-domain SNA network the overhead is intolerable!

Figure 1



Such a network, with a host response time of one second and a network response time of three seconds, would add 75 percent to the user's response time for each additional CICS transaction that must be "started" by a previous transaction rather than using XCTL. What could have been a four second response time became a seven or ten second response time as shown in Figure 1. This quickly results in user dissatisfaction. This increase in response time could be termed the "service cost" of electing a CICS/DB2 implementation.

NETWORK OVERHEAD AND PERFORMANCE

Along with the response time impact of the BID described above, additional network traffic is added. Even though the SNA BID generates a relatively small amount of network traffic in relation to an inquiry, some capacity is still being consumed for no productive purpose. Also, some line band-width is being consumed which could otherwise be used for polling, thus increasing the wait-for-poll component of response time. What is worse, if a company is being charged for network traffic, these extra BIDs are now costing you money and buying you nothing.

Of greater impact are any performance problems that may exist in a given network configuration. Every problem will likely generate twice the impact it might otherwise cause. Every time the BID is transmitted, it will encounter the same network delays (poll delay, queuing, line errors) that normal data traffic encounter. So, if a network configuration bottleneck was degrading response time by one second, the extra BID would now cause the degradation to total two seconds. Network problems have been magnified.

USER RESPONSE TIME MEASUREMENT

User response time measurements are a fundamental part of IBM's network management methodology. Computer Center management and User management are no longer satisfied with only internal host response time reports. User response time reporting has come of age.

However, some interesting complications arise when one attempts to collect and report user response time measurements of pseudo-conversational CICS/DB2 transactions. Lets consider why this is so.

To obtain user response time measurements, software monitors have traditionally used the SNA definite response (DR1) feature. This facility, started by making table changes in CICS, requests that the user's terminal positively acknowledge the receipt of every transmission sent to it. Doing so allows software monitors to measure quite accurately, the user's total response time. The definite response has little impact on CICS performance since nothing critical is waiting for the definite response to return.

Starting definite response in a pseudo-conversational CICS/DB2 environment will add another round trip of the network to the round trip already being made by the BID.

Recall earlier how the second CICS transaction had to wait for the positive response from the BID request before execution began. Now, with CICS definite response started, the first End Bracket transmitted will request a definite response too! This positive response must be received before the BID can be transmitted. The BID will then be sent and must still wait as before. The net result is now two full round trips of the network must be made between transactions one and two. All of the waiting described here is part of the user's response time. Therefore, the use of definite response becomes intolerable. We are degrading response time just to be able to measure it.

What is needed is a means of using the definite response function from outside of CICS, in VTAM itself. This approach immediately rules out IBM's Network Performance Monitor (NPM). The only software monitor that supports this type of operation at present is Netspy from Duquesne Systems. The VTAM activated definite response capability has been developed for Mazdamon and has been successfully tested by the author. But, Mazdamon's new owners, Computer Associates, have chosen not to release this critical feature at this time. Therefore, Mazdamon, as it exists today, is not a realistic alternative for a response time monitor in a CICS/DB2 environment.

By dynamically switching on definite response in VTAM, the second CICS transaction does not have to wait for the End Bracket's positive response to return because CICS knows nothing about the definite response request. So, we once again have measurement at an acceptable overhead cost.

Another problem created by this CICS/DB2 incompatibility arises in how some monitors determine when a "user" transaction has ended. Traditionally, many monitors have used the End Bracket to signal the end of user transaction in a 3270 environment. However, as we have noted earlier, pseudo-conversational CICS/DB2 transactions send at least two End Brackets in each user transaction. Therefore, such monitors will erroneously report two transactions, one short and one medium, rather than one long transaction.

NPM interprets the end of a user transaction as occurring when the Last in Chain, Last in Segment, Change Direction, or End Bracket bit in the last PIU is received. However, these SNA indicator bits, except for the Change Direction indicator, have been observed in the PIU that sends the End Bracket for the first CICS transaction. Therefore, the NPM monitor is effectively useless in the CICS/DB2 environment.

IBM's Response Time Monitor (RTM) functions somewhat differently than NPM. The NLDM/RTM parameter RTDEF allows the user to define what event should signal the end of a user transaction. The CDEB and KEYBD options will not work for reasons described above. These parameters tell RTM to end the response event when Change Direction or End Bracket bits are sensed or when the user's keyboard is unlocked (remember End Bracket unlocks 3270 terminals).

The default for the RTDEF keyword is FIRST. This parameter tells RTM to end the response event when the first character of the reply from the host, that will modify the presentation space, arrives at the user's terminal. This option should work, but has not been tested by the author. The assumption here is

that between transactions one and two, the application program does not send out any intermediate messages to the user's terminal such "Your request is in progress. Please wait." Such an intermediate message would fool this RTM option and provide erroneous response times. Also note that "first character" does not fit the usual definition of response time any may vary significantly depending on the length of time required to display the entire response.

Though not tested by the author, the developers of Netspy claim their product is also able to determine the proper end of a user transaction. The bottom line is that selection of the correct software monitor is a prerequisite to user response time reporting in a CICS/DB2 environment.

PROBLEM DETECTION

To help readers to discover if they have the same problem, we will describe our measurement approach. When response time was not meeting expectations, an analysis was performed on performance of CICS and DB2. After running out of ideas for tuning the host environment, we decided to review the network activity.

To analyze the network traffic in detail, we needed a path information unit (PIU) listing. We used Mazdamon to capture and produce these listings. However, a VTAM buffer trace will produce the same results. The goal was to list the request/response header (RH) and request/response unit (RU) information.

To make the test results as easy to interpret as possible, a series of poor performing test transactions were selected. The same type of terminal being used by our users was used for the test just in case the terminal was having some kind of effect on performance.

After executing each of the test transactions is a specific documented sequence, a list of all of the PIUs was produced. The basic PIU flow observed is shown in Figure 2. As can be seen, an End Bracket was sent in PIU number 0097. Because of the EB, CICS reacquired the terminal with a BID in PIU number 0098. From the time stamps, it was observed that CICS waited for the positive response to the BID PIU, 0098, before continuing processing. The first clue to a serious problem was found!

Figure 2

SNA PIU Flow for CICS/DB2 Transactions

PIU Direction	Seq. Numb	Req/Resp. Header	Req/Resp. Unit	PIU Function Description
Inbound	001F	03 90 A0	7D-ENTER	User hit ENTER BB and CD received
Outbound	0097	03 91 40		End Trans #1-EB sent
Outbound	0098	40 80 00	C8	BID Request sent
Inbound	0098	CB 80 00	C8	Positive Response to BID received
Outbound	0099	02 90 80		BC/BB and data sent
Outbound	009A	01 90 00		EC & last data sent
Outbound	009B	03 90 40		EB sent a 2nd time

When checking the PIU flow in your installation, the key PIU to look for is the BID PIU, number 0098 in Figure 2. The presence of the BID in a 3270 environment, indicates you have this CICS/DB2 problem. If you find that this problem does exist in your installation, the next section discusses IBM's recommended solution, the impact of IBM's recommendation and additional follow-up information.

IBM's RECOMMENDATION - A SOLUTION?

When this problem was first detected in 1985, local IBM SEs were contacted and questions were submitted to IBM via their Equal System. In response, IBM said that if two programs are separate load modules operating under a single CICS transaction, you would need to bind the DBRMs for each program into a single DB2 plan, and this is the plan that would be specified in the CICS RCT. Since the severe performance impact of using large Plans was fixed in DB2 Release 2, IBM suggested this was a reasonable solution to the problem.

While this solution can help to reduce the problem, it by no means eliminates the problems described above and creates several more. In an installation where user productivity and flexibility was one of the primary application design goals, users were given the ability to "jump" from almost any system to any other system. From a personnel system screen, a user can switch to the accounts receivable system or to any other system available. This means to apply IBM's recommendation, the DBRMs for every load module from almost every system would need to be bound together into a few immense DB2 plans. Consider the colossal change control problem this would create.

Also consider, should a small change to any load module or SQL code in any part of a system be in error, DB2 will invalidate the entire plan. Now, instead of losing a single function of a system, one could potentially lose an entire system. In addition, a simple change that required a rebind of a plan, could be very complex to implement. Remember that every time RUNSTATS is executed, it stores data in the DB2 catalog. This data is used at bind time by DB2 in the extremely critical data access path selection process. If the last time RUNSTATS was run, the data base was in a less than optimal condition, the entire data base accessed by the system being changed may need to be reorganized before the rebind is accomplished. If this is not done, the data access path selection process may not pick up data base indexes or decide to perform tablespace scans--both of which can severely impact response time. Thus significant exposure and effort are required for each application code change or any other change requiring a rebind.

The complexities involved in tracking which DBRMs to combine with which load modules, modifying compilers to combine the correct load modules, and testing these combinations all serve to increase the complexity of application software maintenance and the chances for error. This is an absolutely intolerable condition with today's ultra high availability online systems.

In short, the massive binding together of DB2 plans is a "band-aid" solution to a serious problem. In a system of strategic importance

to both IBM and their customers, a more robust approach is required.

As noted above, this problem was first reported to local IBM SEs in 1985. However, little progress toward a resolution was made until March, 1987 when a PASR was finally created on this problem. The PASR was assigned number RKBU110. The title of the PASR is "ALLOW XCTL BETWEEN CICS TRANSACTIONS WHEN USING DB2 PLANS." The description reads as follows: "Need to be able to go from one DB2 plan to another within a single executing CICS task, just as one is able to go from one application program to another program within a single task."

In July, 1987, a request was made for the status on the PASR. The following is the response received from an IBM Senior Product Administrator on the DB2 DB/DC Product Management team: "Regarding PASR RKBU110, your understanding is correct in that development is fully aware of the situation and is working on resolving it. We do not, at this time, have a solution but have taken this PASR as INPUT TO PLANNING" (emphasis theirs).

As of September 1987, no additional information has been received from IBM. If this problem has application in your installation, the author recommends you ask your local IBM SE for a status of PASR #RKBU110.

CONCLUSION

In all fairness to IBM and all others involved in this problem, it is one of those that would have been most difficult to prevent or detect. The complexity of this problem spans the internal architecture of DB2, the internal architecture of CICS and SNA communications protocols. Each of these areas represents a lifelong career to really gain an in-depth understanding. As a result, few people have an in-depth working knowledge of all three. This fact slowed down our ability to communicate the problem to others. In the words of one CICS expert the author spoke to, "I never would have thought of that problem. But now that you describe it in detail, it all makes sense."

In any event, the author is sure that the problem is now being energetically worked on by IBM. Considering the strategic importance of DB2 and CICS to both IBM and their customers (especially those installations that made early commitments to use DB2 in full-scale production), it would seem reasonable that a resolution will not be too long in coming. DB2 Release 4 maybe?

REFERENCES

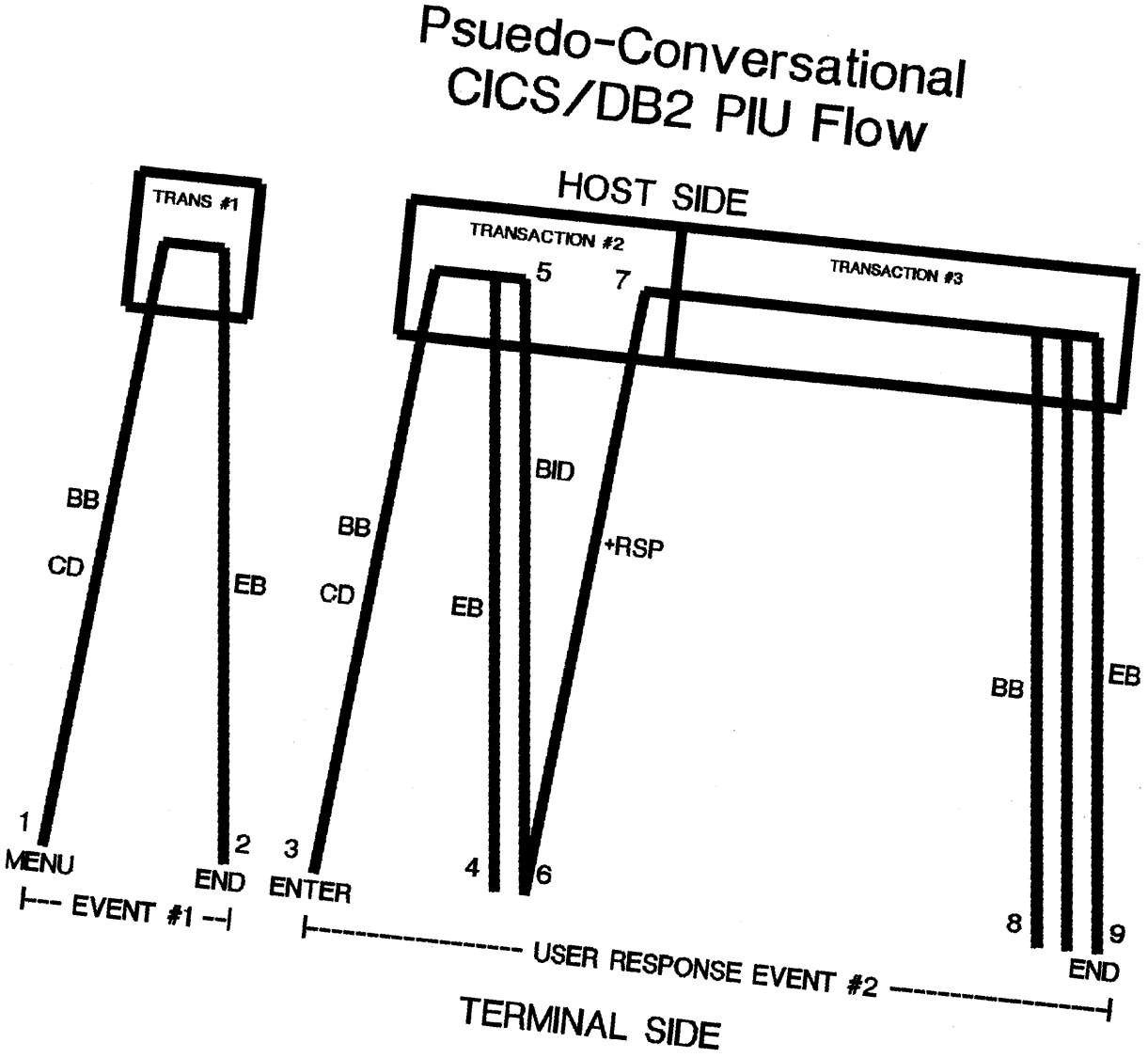
[1] Bozman, Jean S., "IBM to Users: You Should Migrate to Relational DBMS's by 1990", InformationWeek, March 31, 1986, p. 12.

[2] Shane, Jason, "An Approach to Remote Response Time Management Reporting", CMG '86, 1986.

[3] SNA Technical Overview, IBM, GC30-3073-1,

[4] SNA Format and Protocol Reference Manual: Architectural Logic, IBM, SC30-3112-2

Figure 3


Psuedo-Conversational
CICS/DB2 PIU Flow
HOST SIDE
TRANS #1
TRANSACTION #2
TRANSACTION #3
5
7
BB
CD
EB
BID
+RSP
1
MENU
2
END
3
ENTER
4
6
8
9
EVENT #1
USER RESPONSE EVENT #2
TERMINAL SIDE

MODELLING CICS SYSTEMS: THE PERFORMANCE IMPACT OF MRO IMPLEMENTATIONS AND DATA BASE CONSIDERATIONS

Ellen M. Friedman
E.M. Friedman, Inc.

ABSTRACT

Today, CICS systems are more complex and require more resources than ever before. The process of measuring and modelling CICS systems has also become more elaborate, a result of recent architectural enchancements in Inter-system Communication (ISC), inter-region communication and particularly with Multiregion Operation (MRO).

As performance analysts, we are concerned with measuring CICS performance and determining its resource usage, regardless of the particular CICS design architecture. As capacity planners, we are concerned with anticipating the impact of alternate implementation strategies on future on-line system performance.

This paper reviews the various CICS architectural options including: MRO, ISC and database interfaces, and presents the relevant performance measurements available to quantify CICS resource utilization under each of these implementations. A general methodology for representing CICS workloads in analytic models is presented, as well as a specific workload characterization and modelling methodology for an MRO system with several data base interfaces (DL/1, Adabas). Several examples illustrating the applicability of these techniques are included. A case study comparing projected MRO usage with actual measured data will also be discussed during the presentation.

Introduction

CICS provides an environment that allows for considerable architectural flexibility in the design of DB/DC applications. As a result, the number of installations utilizing CICS has grown significantly in recent years. CICS systems can support a wide variety of customized on-line applications encompassing multiple subsystems which are capable of communicating across a large SNA network. The variety of design and architectural features are numerous. CICS provides file control capabilities to support various IBM data access methods and data base products and can interface with other non-IBM data bases including IDMS, Adabas and DATACOM/DB. Furthermore, CICS systems can communicate with other CICS systems and data base regions, within the same processor complex, as well as across TP links to other CPU's.

The modes of intercommunication available with CICS include Intersystem Commmunication (ISC) Multiregion Operation (MRO) [ref.1] . With Intersystem Communication, one CICS system can access files and data bases from another region (CICS or IMS) in a connected system via SNA links. The Multiregion Operation (MRO) feature allows for communication between multiple CICS regions executing within the same processor complex. The use of data base handlers such as DB2, Adabas, or IDMS is an example of Interregion Communication between CICS and a particular data base region.

The use of MRO and ISC has increased significantly in recent years. There are numerous reasons why a CICS installation might consider an MRO or ISC implementation. For example, using multiple CICS regions, applications can be designed to control resource accessibility, to maximize on-line availability and to ensure proper security measures for sensitive applications. In addition, transactions can be grouped based upon response time criteria, permitting consistent performance to time-sensitive requests (e.g., ATM banking transactions). In a multiple CPU processor complex, several subtasks can execute concurrently. As a single address space, CICS cannot take full advantage of the processor complex. If the application were partitioned into several regions, overall throughput could be improved. Other advantages offered for multiple regions include virtual storage constraint relief and the ability to share resources, such as terminals and files, among several CICS regions.

Following Release 1.5 of CICS, there have been significant enhancements and improvements in the ability of CICS to communicate with other subsystems. Release 1.6 offered many performance improvements in MRO, and the most recent Release 1.7 is expected to provide even better performance and improved monitoring capabilities. As a result of these improvements, many CICS users have considered redesigning their applications to process in multiple CICS regions. Since there has been an increasing number of installations contemplating the use of the MRO and ISC features, the concern for modelling this environment has increased as well.

The remainder of this paper will address the data analysis and modelling issues relevant to the prediction of future system performance for CICS systems utilizing MRO or ISC architectures. In the first section, we will present a brief overview of the interregion communication facilities and discuss the data sources and analysis issues relevant to measuring and modelling CICS MRO systems. Secondly, we will discuss the general techniques for evaluating the performance of CICS systems utilizing MRO with analytic modelling tools. As part of this discussion, we will present a methodology for determining the performance impact of converting an application that is currently executing in a single CICS region to one processing in multiple regions. We will give specific examples of modelling existing CICS systems which utilize MRO. During the presentation, we plan to review a case study to illustrate the techniques and highlight some of the performance considerations in implementing the MRO feature.

Background

Overview MRO and ISC

The CICS intercommunication facility includes CICS function request shipping, transaction routing, distributed transaction processing and interregion communication. Intersystem Communication (ISC) provides CICS function shipping and distributed transaction processing, enabling an application runnning in one CICS system to access files and data bases and to start processing on connected processors. With the Multiregion Operation (MRO) facility, transaction routing, function shipping and distributed transaction processing functions are all provided [ref.4,5].

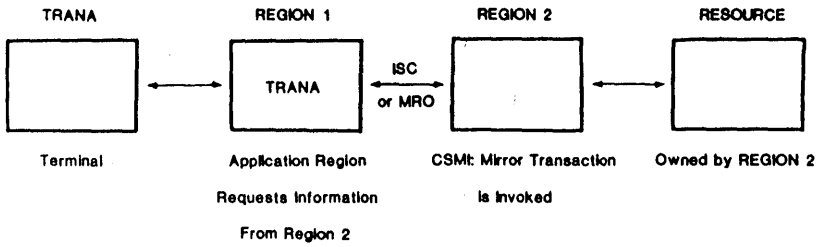


Figure 1. Function Shipping

Function shipping allows an application program to make a request for a resource in a different CICS system, have that request shipped to the system controlling the resource and receive the information back without knowing where that resource is. The CICS system can be in a different processor or a different region [Ref. 4,5].

With transaction routing, a terminal in one CICS system can initiate a transaction in another CICS region. Distributed transaction processing allows communication between transactions executing in either an MRO connected region or an SNA-connected system. [Ref 1].

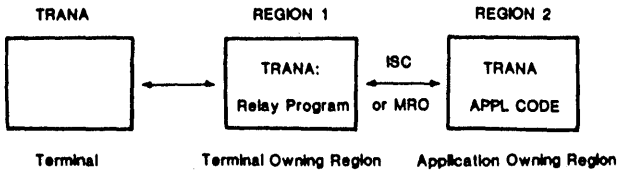


Figure 2. Transaction Routing

In order for a region to utilize MRO for function shipping or transaction routing, it must identify itself to the other regions and define the connection to those regions with which it wishes to communicate. The connection between systems is accomplished by coding system entries in the terminal control table (TCT) identifying all remote systems with which that system is able to communicate. The implementation of function shipping requires that all remote files, remote transient data destinations and remote temporary storage queues be defined in the appropriate tables. Transaction routing permits an operator to enter a transaction at a terminal and have that transaction routed to another CICS region for execution. This is accomplished by defining the transaction as remote in the program control table (PCT) and identifying the region in which the application code corresponding to that transaction resides.

Data Bases

CICS allows access to data held within standard OS data sets, areas within CICS and in data bases. Information can be accessed from standard data sets using the CICS file control facility with VSAM, ISAM or DAM files. The information can also be accessed from both hierarchical and relational data base systems, such as IMS/DLI and DB2.

Data Analysis

Performance Monitoring and Analysis

In terms of measuring CICS performance, various statistics are available with both IBM and non-IBM monitoring products. The performance data is typically summarized for each transaction, or transaction/terminal combination and includes some information concerning the usage of memory, I/O and CPU resources. For the most part, CICS monitors and report writers simply collect the data without performing any analysis. Performance analysts, however, must concern themselves with resource usage by region and by transaction, regardless of the particular CICS design architecture or data base product used.

Performance data is recorded separately for each region. Depending upon the architecture, transaction response time recorded in the local region may include the time it spent processing in a remote region. This typically occurs with remote data base usage, transaction routing and function shipping. The resources consumed by the transaction in the remote region will normally not appear as part of the CICS performance records (The exception is if event monitoring points are used for data base requests.). As a result, it is up to the analyst to determine the component parts of transactions and reconcile the amount of resources utilized in remote regions. This analysis is particularly important for job accounting, service level management and in the development of analytic models to predict future system performance.

Function Shipping

The manner in which performance data is collected and reported by various monitor facilities depends upon the particular architecural implementation that is used. Within the CICS monitor facility data, mirror transactions initiated by function shipping requests appear as CICS CSMI transactions. The types of function shipping requests include temporary storage, transient data, interval control, file control and DI/I. The specific CSMI transaction name coded depends upon the type of request. However, in Releases prior 1.7, no information that indicates the relationship between the two communicating regions is recorded. As such, there is no direct method of determining the resources consumed by a transaction in a remote region when the mode of inter-region communication is function-shipping. In this case, since no terminal identifier is recorded in the receiving region, transaction resource usage can only be reconciled based upon the date/time stamp. In a busy system with several regions, the accuracy with which one can pinpoint the resource usage is questionable.

With Release 1.7, the VTAM Applid (a 20-byte field) and a UOW field (8 bytes), the name by which this unit of work is known within the originating system (assigned at attach using an STCK-derived token), will be passed. The STCK time stamping ensures the uniqueness of each entry and facilitates mapping the originating systems request for service with the actual usage in the processing region [ref.2].

Transaction Routing

In a typical scenario, transaction routing takes place between a single terminal- owning region (TOR) and many

different application-owning regions (AORs). The terminal region is responsible for servicing and routing transactions to the appropriate application region. There are two different types of transaction routing which exist: explicit routing through use of the "routing transaction" CRTE and routing through the use of a "relay transaction". In the first case, the operator signs onto a terminal-owning region and explicitly enters "CRTE" followed by the SYSTEM ID of the CICS region with which he wishes to establish a routing session. In the second case, a terminal operator enters a transaction code which is associated with a remote region. The transaction which is attached executes the CICS "relay program" and therefore is referred to as the "relay transaction". The major difference between the two types of transaction routing is their implementation and the resulting manner in which performance data is recorded within the CICS monitor facility (CMP).

Regardless of the implementation, the analyst needs to be able to attribute the time spent and the resources used within the terminal-owning region to total transaction resource consumption and response time. Where the routing information is explicitly coded in the program control table, the task of correlating transaction resource usage across regions is that much easier. In this case, the same transaction ID will be reported in both the receiving and sending regions. As a result, the transactions usage of resources in both the terminal and application owning regions can be identified. When the CRTE transaction is executed within the terminal-owning region, no data will be recorded that identifies the transaction to be routed to an application region. In this case, the analysis effort is slightly more complex. To correlate resource usage in the terminal-owning region with transaction response time in the application-owning regions, the analyst must compare terminal and operator identifiers, and date/time stamps between the two regions. Table 1 illustrates the analysis.

Data Base Usage

With the CICS monitoring facility (CMP data), transaction usage of CICS VSAM data bases and DL/1 data bases are automatically recorded in the performance records. Transaction usage of DB2 data bases can be obtained from the DB2 accounting data which is written to SMF as a type 101 record [ref. 7,8]. By defining event-monitoring points in user application programs, the number of file calls and their associated timings can be obtained for all other data base products [ref. 3].

Specifically, there are 255 user-defined counter and clock fields available with CMP. To make use of these counters and clocks, event monitoring points (EMP's) are coded at the desired location within the application program. In addition, the EMP's must each be uniquely identified by a number and an entry corresponding to that number must be coded in the monitor control table (MCT) [ref. 3].

By coding EMP's, the number of file control calls and their associated timings can be recorded in the performance record for each transaction that utilizes interregion communication to access data from another data base region, such as Adabas. Table 2 illustrates the usage of user fields.

Table 1: Example of Transaction Routing
Performance Data including CRTE

(In descending order of START TIME)

SYSTEM	START TIME	END TIME	TRANID	TERMID
CICS1	10:32:17.22	10:32:18.76	X100	ABCD
TOR	10:32:17.21	10:32:18.77	CRTE	ABCD
CICS1	10:31:59.52	10:31:59.57	Y100	ABCD
TOR	10:31:59.51	10:31:59.57	CRTE	ABCD
CICS1	10:31:38.13	10:31:40.44	X100	ABCD
TOR	10:31:38.13	10:31:40.45	CRTE	ABCD
CICS1	10:31:30.02	10:31:30.09	Y100	ABCD
TOR	10:31:30.01	10:31:30.09	CRTE	ABCD

Table 2: Example of User Clock and Count Field
Adabas file usage with Event Monitoring Points

(Average values are recorded and
timings are reported in seconds)

TRANID	N	RESPONSE	CPU	AMCT	USERCLK	USERCNT
BAG	5	.53	.024	0	.304	4.8
CRTE	28	.63	.008	0	0	0
GAB	419	.91	.057	3.4	.182	2.8
MACK	6	2.18	.348	253.2	0	0

NOTE: This example illustrates that any transaction can issue file I/O to: CICSVSAMdatasetsand other data base systems or not issueanyI/O whatsoever, or some combination. USERCLK represents average data base service time; USERCNT represensts average number of data base requests.

Modelling Considerations

Overview

Analytic modelling tools can be successfully employed to predict future workload performance for MVS systems. The basic inputs to these tools are descriptions of the hardware, software and workloads operating in the environment. In describing the workload, one must include CPU and I/O resource usage information, define the amount of real memory available for the workload and specify the workloads CPU priority. Memory usage is represented to the model in terms of maximum multiprogramming levels (MAX-MPL), attained multiprogramming levels (ATT-MPL) and measured page/swap service times.

In a previous paper we outlined a methodology for modelling CICS systems. We indicated that there are some particular issues to be considered when modelling CICS systems [ref. 9,10]. Specifically, the issues include paging delays, enqueue delays, single address dispatchability in a multi-processor environment and the ramifications of modelling heterogeneous CICS workloads. In addition, we discussed the techniques for forming multiple CICS workloads in an analytic model. We will not repeat the general techniques for parametrizing a CICS system within the analytic model construct, but rather we will focus on the particular issues involved in modelling interregion communication between CICS systems. As a special case of interregion communication, we will also discuss modelling the interface between CICS and other data base regions.

To model a CICS application executing in multiple regions, the following steps must be performed:

1) Identify the relationship between the regions (e.g., Terminal-Owning Region, Application-Owning Regions or Sending and Receiving regions).

2) As an initial step, place each region in a separate workload. (In this case, we are assuming that each region appears as a distinct performance group in the IPS.)

3) Determine the CPU priority and MAX-MPL for each workload/region.

Given that you have identified the relationship and communication path between the regions, the next step is to obtain the data required to parametrize the specific resource requirements of each of these regions within the analytic model.

Iterative Modelling Technique

The method employed for modelling both MRO and ISC implementations will utilize "iterative modelling" techniques. This technique can be used to anticipate the performance impact of converting to MRO and to model an existing MRO implementation [ref. 11].

With both function shipping and transaction routing, a transaction in one region (sending) causes a transaction in another region (receiving) to be initiated. The transaction in the sending region must wait for the task to be completed in the remote region. As a result, the response time for the initial transaction, includes the time necessary for the receiving region to process the work requested. However, the resource profile developed for the sending region only accounts for resources that have been utilized by and charged to that region. Since RMF/SMF information is recorded at the performance group level, average resource profiles constructed for a particular CICS application (based upon an analysis of this data) would not reflect the resources that application consumed while executing in a remote system or region. Therefore, any model based upon this particular resource profile description, which only accounted for locally consumed resources, would be inaccurate.

Utilizing the technique of iterative modelling, one can include the processing time for the transaction in the receiving region as input to the calaculation of the response time for the sending region. This is accomplished by defining a delay server representing the processing time occuring in the receiving region. (A delay server is defined as a server at which no queueing occurs and as such, it is just a method of adding a constant value to the model). This delay server is then added as part of the resource profile for the sending region [ref. 11].

In summary, using iterative modelling techniques, two regions communicating via MRO or ISC implementations can be analyzed as follows:

1) Begin with a model containing at least two workloads, corresponding to the sending and receiving regions.

2) Execute the model and determine the response time for the receiving region.

3) Define a "delay server" in the sending region which is equal to the response time of the receiving region.

4) Evaluate the model. If response time for the receiving region has changed since the last iteration, change the value of the "delay server " in the sending region to equal the new calculated response time.

5) Repeat the process until the calculated response time for the receiving region approaches a consistent value.

The next few examples illustrate some of the specific considerations and parameters necessary to model the performance impact of converting to MRO. They also demonstrate the techniques for modelling an existing MRO system and highlight the methods necessary for modelling a CICS interface with another data base system.

Modelling an MRO conversion

This first example illustrates the modelling techniques for anticipating the impact of converting to MRO. In the original environment there are two application-owning regions, each of which is responsible for its own terminal control. The installation has considered adding a single terminal-owning region to handle all terminal interactions and route the transactions to the appropriate AOR, as necessary. By maintaining a single copy of the TCT, some virtual storage relief is expected. In addition, under an MRO implementation, CICS end users will only need to log on to a single region and still be able to access information from either of the two AOR's.

Converting to MRO or implementing ISC will result in an increase in CPU overhead to support interregion communication. If the implementation requires an additional region, an increase in the demand on real storage demand can also be expected. One can estimate the additional amount of real memory required from guidelines reported by IBM in "MVS/XA Processor Storage Estimates and Performance " [ref.6]. The guidelines presented suggest that the base storage requirements for a production CICS/VS region (under XA), is in the 1.5 to 2.5 megabyte range. The amount of real memory required depends upon the number and size of resident programs, the amount of concurrently executing transactions, and the average storage requirements for each transaction.

Of specific concern is the prediction of CPU utilization and response times for the new MRO environment, with a single TOR servicing each of the AOR's.

Modelling Assumptions

1) The majority of the terminal control activity occcurs in the TOR causing a significant reduction in the TC portion of CPU time utilized by each of the AOR's in the current environment.

2) The TOR will utilize TC and KC facilities. Determining the total CPU time that the TOR will consume requires knowledge of the average pathlengths for Task Control and Terminal Control for a Terminal Owning region (under MRO). This is expressed in Equation 1.

Model Inputs

1) Build a model with CPU time for the TOR workload defined as above.

2) Recalculate the CPU time for each AOR by subtracting out the portion of TCB attributable to terminal control.

3) Determine the MAX-MPL for the TOR and each of the AOR's.
(The value will depend upon the size of the Dynamic Storage Area, and the amount of real memory available for each transaction. Discussions of memory modelling as they pertain to CICS can be found in reference 10 and general memory modelling techniques are discussed in reference 12).

4) Run the model.

5) Use the iterative modelling technique as defined above, with delay servers defined for each AOR equal to the average response time for the TOR.

Modelling the performance impact of converting to MRO when the mode of interregion communication will be function shipping will follow the same principles.

Modelling an Existing CICS/MRO System

CASE I: MRO/Transaction Routing

This example illustrates transaction routing in an environment with two application owning regions (AOR) and a single terminal owning region (TOR). All CICS users sign onto the TOR and transactions are routed to the designated AOR. (See figure 2) Since the TOR can also execute application code and contain resources which are accessible to other regions, one must divide the corresponding TOR workload into two workloads. The first workload will contain the transactions which serve as the communication vehicle to the other two AOR's. The second workload will contain the remaining transactions. The same division of transactions (those communicating with the TOR and those not) needs to be applied to each of the AOR's.

The two methods of implementing transaction routing are:

A) Using the relay transaction
B) Using the routing transaction, CRTE

The advantage of using the relay transaction is that one can readily map the movement of transactions across the regions. The same transaction ID will usually be specified in both the terminal and application-owning regions.

A) With the relay transaction-

1) Divide the TOR into at least two workloads- one corresponding to each AOR
2) Divide the AOR 's into those transactions communicating with the TOR and those not.
3) Specify the MAX-MPL and CPU priority for each workload
4) Determine the resource profiles for each of the new workloads.

B) With the CRTE transaction-

1) You can separate the resource usage for the CRTE transaction, on behalf of each AOR, by the terminal/operator identifiers and date-time stamp generated for each CRTE transaction. (The terminal and operator ID for each AOR will correspond to that generated for specific CRTE transactions.)

Repeat steps 2 through 4 in A) above.
Alternatively, leave the TOR as a single workload and determine its average processing time (without waits).

2) Add this time as a delay server to each of the AOR's.

3) Adjust the MPL's for each of the AOR's to represent the maximum number of tasks that the TOR could execute concurrently on behalf of that AOR.

4) Refine the process and perform "what-if" analysis utilizing iterative modelling techniques.

Predicting the Impact of Workload Growth

To anticipate the effect of additional volume on system performance, one must first determine which transactions are expected to increase. If an increase in volume is expected for only one of the applications, then the model inputs need to reflect this. To facilitate this, two new workloads should be constructed based on a decomposition of the terminal-owning region. Each of the newly formed workloads would correspond to the activity performed on behalf of each of the individual application owning regions. As a result of forming these workloads, one can directly increase/decreasse the volume for the TOR workload corresponding to the portion of the terminal owning region which is expected to change on behalf of that application. Without separating the TOR in this fashion, we would not be able to determine the additional overhead that the application would cause in the terminal owning region.

Case II: Function Shipping

The facility with which one can model a function shipping implementation of MRO or ISC is dependent on the data available and the ability to correlate resource usage in a remote region with transaction response time.

Unfortunately, with Release 1.6 of CICS there is no way to identify the region that initiated a function shipping request. Therefore, using the performance

$$\text{Total CPU Time for TOR} = \frac{\text{KC pathlength} * \#\text{ transactions}}{\text{processor speed}} + \frac{\text{TC pathlength} * \#\text{ transactions}}{\text{processor speed}} \quad (1)$$

Note: The pathlengths were developed by BGS systems using representative benchmark studies.

monitor data, one cannot readily build typical resource profiles for workloads which use MRO or ISC for function shipping. Since the resource profiles will not necessarily be representative, one cannot expect that the models developed to be accurate. However, all is not lost as one can always make some intelligent estimates.

For example, if only two regions are involved, the CSMI transactions will be recorded in the region containing the resource of interest. Prior to Release 1.7, we were not able to identify the initiating transaction ID. However, we would be able to isolate the CSMI transactions into a separate workload and employ the iterative modelling procedure to determine average response times and resource usage for each of the two communicating regions. (In Figure 5 the request handlers would be the CSMI transactions.)

If multiple regions utilize function shipping, it will be extremely difficult to trace the CSMI transaction recorded in one region back to the region that requested the resource. One can attempt to map the CSMI transaction to the initiating region by analyzing the date/time stamps for all transactions executing in these CICS regions.

With Release 1.7, a date/time stamp and region and transaction identifiers will exist in the performance records to facilitate mapping the path of a transaction through the various regions within which it executes.

Regardless of the release level of CICS, if one can identify the amount of resources that a particular transaction consumes within each region, then one can model the system environment utilizing the techniques presented for transaction routing.

The following example illustrates the procedure.

CICS Function Shipping Modelling Example:

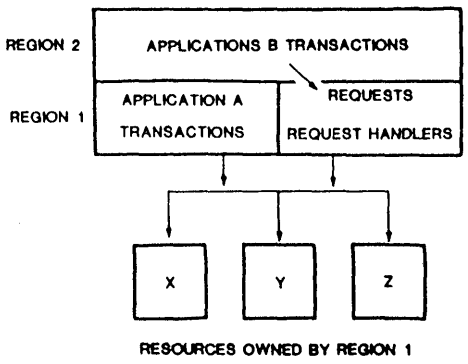


Figure 3.

CICS Region 1: Owns resources X,Y,Z and APPA requests X,Y,Z directly

CICS Region 2: APPLICATION B sends request to Region 1 to access X,Y,Z

The model would contain at least 3 workloads defined as follows:

Workload 1 - corresponds to APPLICATION A

Workload 2 - corresponds to Request Handlers in Region 1 (note: We have assumed that sufficient data would exist to facilitate the identification of these transactions.)

Workload 3 - corresponds to APPLICATION B

Modelling Procedure:

1) Workload 1,2 would be specified as processing in the same (or shared) domain.
2) A delay server, representing the amount of time spent by workload 3 waiting for the request handler to process the request would be required.
3) Iterative modelling technique would be employed.

Function Shipping-Modification Analysis

To determine the performance impact of workload growth, the following steps should be implemented:

1) Adjust the transaction volumes in accordance with the projections obtained.
2) Recompute the Delay servers.
3) Reevaluate all response times.

CASE III. Modelling a CICS interface with another Database

If both CICS and Batch are performing inquiries and/or updates against the database, one must first identify the amount of activity and resources consumed by each of these systems. By analyzing the log tape for most data base products, one can determine the run unit responsible for the activity. For example, with Release 2 of the DB2 product, the data recorded in the accounting records will identify whether the resulting file activity was caused by TSO, Batch or CICS. These records will also contain the number and type of SQL calls performed by each user authorization ID (e.g. signed on user, terminal or transaction ID). Once the amount of activity is reconciled between the various regions (i.e., how much was consumed by BATCH vs. how much was used by CICS), the specific representation of resource usage can be input to the model.

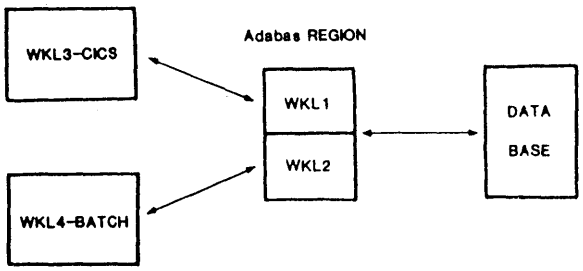


Figure 4.

To facilitate modelling changes to any of the three systems, BATCH, CICS or Adabas, we recommend starting with a baseline model containing 4 separate workloads. This will also enable the analyst to model regions with differing CPU priorities and MPL's.

The specific modelling details are as follows:

1) Form four workloads, where the resource profiles of workloads WKL 1 and WKL 2 are formed as a result of apportioning the original single Adabas resource profile into two distinct profiles according to the number of ADABAS calls issued by each region.
2) WKL1, WKL2 (Adabas workloads) should be placed in a common/shared domain.
3) Determine MAX-MPL's and CPU priorities for each workload (where the MAX-MPL and priority is the same for both Adabas workloads)

4) Add a delay server to the BATCH workload. The value should be equal to the average amount of time required for the Adabas region to perform the file activity requested by BATCH. Therefore, the delay server will be equal to the response time for WKL3 as defined above.
5) Add a delay server to the CICS workload. The delay server will be equal to the response time computed for WKL4.
6) Apply the iterative modelling techniques.

To anticipate the impact of workload growth on performance, the transaction volumes for all workloads which are expected to change should be adjusted. Using the iterative modelling methodology, one can recompute the values for the delay servers and determine the response time CPU and I/O utilizations for each of the workloads.

Summary

As a single address space, CICS has limited opportunity for utilizing a multiple CPU complex, such as an AP, dyadic or MP. By splitting a single region into multiple ones, multiple subtasks can execute concurrently to improve application performance in this environment. As a result, MRO and ISC architectures have become much more common.

We have outlined some of the performance and modelling issues which pertain to CICS applications utilizing this architecture. We have reviewed the general methodology for modelling CICS applications and have proposed several techniques for predicting the performance of CICS systems utilizing MRO and ISC. We have indicated that the specific details for analyzing any of the various implementations of MRO and ISC are similar; iterative modelling techniques are employed to determine the total response time for these transactions which consume resources in multiple regions.

The implementation of MRO and ISC will impact performance, there are costs in terms of CPU overhead and real memory. However, depending upon the business requirements, the particular resource requirements for the application, and the system environment in which that application will execute, the decision to take advantage of the MRO architecture may be the most prudent and cost effective design alternative.

Acknowledgment

I would like to thank Ms. B.J. Hoyt of the Health and Welfare Data Center, Sacramento, California, for her helpful suggestions and numerous discussions concerning the performance impact of an MRO conversion.

References

[1] CICS/VS General Information, IBM GC33-0155-1 2nd Edition, December 1982

[2] CICS/VS Performance Analysis Reporting System, IBM SB21-2495-5

[3] CICS/VS Customization Guide 1.6, IBM SC33-0131-1, June 1983

[4] CICS/VS Intercommunication Facilities Guide, IBM SC33-0133-1, June 1983

[5] CICS/VS Multiple Region Operation Facilities and Planning Guide, IBM G320-5902-0, April, 1982

[6] S.N. Allen, G.M. King, MVS/XA Processor Storage Estimates and Performance, IBM Washington Systems Center Technical Bulletin, IBM GG22-9397-01, July 1985

[7] Share 66 Anaheim, Elliot, Don Dallas Systems Center, IBM DB2 Concepts and Facilities

[8] Share 66 Anaheim, Ojala, R.N. CICS/VS and IBM DB2 V1 R2 Performance Considerations

[9] E.M. Friedman, Workload Characterization for Subsystems: A Case Study for CICS/VS, CMG XIV, December 1983

[10] E.M. Friedman, J.L. Rosenberg, Workload Characterization for CICS/VS: The Modelling Implications, CMG XV, December 1984

[11] CRYSTAL/CICS Release 2, BGS SYSTEMS, Inc., August, 1984

[12] Glen Becker, Modelling the Memory Impacts of TSO and Batch growth, CMG XV, December 1984

P.S.* I LOVE YOU. SIGNED TSO
* P.S. = Processor Storage

Bernard A. Lichtig
Russ W. Martindale
Terri L. Parmelee

Amdahl Corporation
Sunnyvale, California

ABSTRACT

This paper will show the results of a series of experiments in uniprocessor and dual processor environments. The experiments were designed to show how the Time Sharing Option (TSO) uses the main storage that is installed on the configuration. 48MB to 256MB of main storage will be used in the physical experiment cycles. The Teleprocessing Network Simulator (TPNS IBM Program Product 5662-262) will be used to provide the scripts for the simulated workloads required to drive the measured processors to saturation.

Logical swapping versus physical swapping is studied to show how the operating system uses main storage. The more main storage, the more the user remains in that storage and is controlled by the think time aspects of a session.

A copy of the presentation slides is contained in the appendices. References to the slides are made in the text and cross-referenced for the reader. Photo reduction of the slides would make it difficult for the reader to interpret some of the values.

These results have been developed through measurements of workloads created by Amdahl personnel. These workloads are only for approximation of true production workloads. Estimation of relative performance for a given workload is dependent on the specific application and operating environment.

INTRODUCTION

In the early seventies, simulation techniques were used to show what would happen in a CAD/CAM environment if the user were kept in main storage instead of rolling the working set out to auxiliary storage devices. The operating system (MVT) did not have a "Logical Swap" feature, but if enough main memory or the region size was large enough, the working set could be placed in a dormant state and then be returned to active status on demand (see [1]).

As more of the operating systems entered into the Multiple Virtual Systems (MVS) environment, logical swap became more of an integral part of the operating scene. In short, the user's working set could be marked logically swapped or dormant. If the main storage occupied by one user was not required by another task, the logically swapped user stayed in main storage. If the occupied main storage were required by some other task, the dormant user could be physically swapped to some auxiliary storage device (DASD). The controlling factor would be the amount of think time that a user was allowed to use before a decision was made to recommend that the user be swapped out.

In the last few years, larger main storage sizes have become available and the logical swap condition takes on a more important role in the Time Sharing Option (TSO) online subsystem. The Amdahl Performance Evaluation Center (AmPEC) located in Sunnyvale, California has conducted a series of experiments in various online subsystems to show the effects of increased storage sizes. This paper will document the results of those experiments in uniprocessor and dual processor complexes that employ main storage sizes of 48MB to 256MB.

METHODOLOGY

In today's complex computer centers the USER is required to submit estimates of requirements for resources. (See Figure 2.) The Capacity Planner uses this set of inputs to plan for the user's needs and the physical equipment that would be required as the current workload grows. In most cases the user's inputs can be + or - 30%. Therefore, the capacity planner's output or plan can produce more questions than answers.

Amdahl's approach is to put the yoke on the vendor and supply what the capacity is of each computer configuration as it completes prescribed workloads. (See Figure 3.) AmPEC has produced a series of documents called Planning for Capacity in TSO, CICS, IMS, and VM/CMS environments (see [2 - 8 incl.]). These studies try to illustrate what is the maximum amount of work the tested configuration can produce using a given workload, then compare and contrast what various configurations can produce in uniprocessor and dual processor environments. It is important to find out what the maximum capacity of the computer is in a totally unconstrained environment. In other words, eliminate any storage constraints or I/O bottlenecks that would keep the processor from running at 100% CPU busy.

What are the users' online terminal requirements? (See Figure 4.) Perhaps it might be 500 TSO users or 1000 IMS users that input a workload to the main central processing unit (CPU). It is not economically feasible to have the 500 or 1000 users available for testing. Nor is it feasible to have the 500 or 1000 physical terminals available for testing purposes. Therefore, in the laboratory, this is simulated with the Teleprocessing Network Simulator (TPNS - IBM Program Product 5662-262). In the AmPEC center this simulated workload is relegated to a driver system. The driver system is the host for TPNS that will simulate the TSO or IMS workload. This driver system is called the System Not Under Test (SNUT). The TPNS simulation is placed on the SNUT so that none of the associated overhead would appear on the system that is being measured. This other, or measured system, is called the System Under Test (SUT). The tested application (TSO or IMS) is simulated as a local network so that none of the associated network delays would enter into the measured response times.

The bridge between the driver system and the system being measured is accomplished by a communications front-end processor. (See Figure 5.) When the simulated requests bridge over to the system being measured, they appear as devices submitting work or transactions to TSO, as an example. Let's say that TSO is the only subsystem on the measured processor. Therefore, the operating system and TSO will be coresident or TSO will be running in a dedicated environment. In the experimental cycle, the number of TSO users will be increased on the SNUT until the CPU being measured is operating at 100% CPU busy. Each TSO user will be assigned a script or terminal session to complete. The user will LOGON, do the required work steps, and then recycle through the work steps. This will be repeated over and over again during a measured period of time. As more users are put into the operating environment, it is assumed that the workload remains homogeneous (more of the same).

In order for the device's transaction to reach the processor, the device must be connected to a data path or channel. (See Figure 6.) Each tested configuration can have a different set or maximum number of channels attached. The channel paths are important requirements to keep any I/O contentions from forming. Too few channels for the workload can add wait time to each transaction. This additional time is additive to the request's overall response time. On the other hand, a configuration's maximum channel capability is an important future growth input parameter.

Lastly is the presence of the processor. (See Figure 7.) This is the SUT. The experiment is designed to try to find out what the maximum capability is of this processor with a workload that can be increased until processor saturation (CPU busy = 100%) is reached. The striations on the channel designators will be used to illustrate various amounts of main storage. As device and channel constraints are eliminated, the main storage can be the limiting factor. When main storage becomes the limiting factor, increased paging and swapping functions can lead to increased I/O demands. This can produce a new set of I/O bottlenecks. Therefore, the size of main storage will be increased as more and more users are added to the workload generation schema. An online subsystem like TSO will use as much main storage as there is present in the operational environment.

Before the experimentation cycle begins, it is important to define some of the terms that are used. (See Figure 8.) Suppose we have a measurement interval of one-half hour. Then suppose the CPU is busy 50% of the time, or 900 seconds. Externally, or to the user's eye, the External Transaction Rate (ETR) is the number of transactions divided by the number of seconds in the measurement interval. Internally, the Internal Transaction Rate (ITR) is the number of transactions divided by the number of CPU busy seconds. Unless the CPU is busy 100% of the time, these are two totally different metrics. When the processor is operating at 100% CPU busy, the ETR and ITR will be equal.

(See Figure 9.) In the first column, the processor is a 5870, which averages 61.51% CPU busy. 14,524 TSO transactions were processed in the 30-minute measurement interval. The ITR is 14,524 transactions divided by 61.51% of the 1800 seconds (or 1107.18 seconds) for 13.14 transactions per second. The ETR is 14,524 transactions divided by 1800 seconds (or 8.08 transactions per second).

In the second column the processor is also a 5870, which averages 98.20% CPU busy. 27,392 transactions are processed. The ITR is 27,392 divided by 98.20% of 1800 seconds (or 1767.60 seconds) for 15.50 transactions per second. The ETR is 27,392 transactions divided by 1800 seconds for 15.22 transactions per second. As the processor approaches 100% busy, the ETR and ITR converge and are equal.

A configuration that is not operating at 100% CPU busy is subject to low system utilization effects. This is an operating system phenomenon that is caused by recursive scanning of the work queue. The software thinks that there is always something to do, and if after the first pass of a no-find situation, it goes through a recursive scan. To eliminate the low system utilization effects, an artificial work request called the BR15(s) job is placed in the processor (two jobs in a dual processor). This is a Branch on Register 15 to the same instruction address and creates a loop that is interruptable when higher priority work is in the system.

Consider this first example of a 5870 and a 5860 operating without the BR15(s). (See Figure 10.) The workload used in this set of experiments was the Short Transaction Workload. This workload simulates a program development complex and represents a higher supervisor state than problem state. In other words, there are many more transactions simulated in the measured period, but each transaction requires minute amounts of CPU. The 5860 uniprocessor used 39.74% of the 1800 seconds or 357.66 seconds to process 8879 transactions. The ITR is 24.92 transactions per second. The 5870 dual processor averaged 32.89% CPU busy, or each processor was busy 296.01 seconds to process 8878 transactions. The ITR here was 29.99 transactions per second.

Therefore, the Multiprocessor Ratio is 29.99 divided by 24.92 or 1.21. This translates to the dual processor doing only 21% more work than the uniprocessor. BAD ROI!

Now set up the BR15(s) to run as the lowest priority job(s) in the work queue. Place the BR15(s) in its own reporting period and performance group so that all CPU time(TCB time) for the BR15(s) can be factored out. (See Figure 11.) The measured processors will be running at CPU busy = 100%. Once the BR15 time is factored out, the same number of transactions now takes 157.50 seconds on the 5870 and 269.00 seconds on the 5860. The transaction counts are virtually the same; ITR on the 5870 is 56.33 transactions per second and 32.97 transactions per second on the 5860. The MP ratio is 1.71. In other words, the 5870 is producing 71% more work. Later in the experimental series you will see the effects of the Long Transaction Workload. In this workload the CPU requirements are weighted with heavier problem program state versus supervisor state. In this set of experiments the BR15 is removed and all real work is present. The MP factor will go to the 1.83 to 1.95 range. This then would be the true MP factor if each processor is running at 100% busy.

UNIPROCESSOR

The experimental series starts with a 48MB Uniprocessor (5860). (See Figure 12.) A LONG transaction workload was created to use up the maximum number of CPU cycles, but still meet certain user service level agreements. A LONG transaction workload is defined as a higher Problem State ratio versus Supervisor State. The average response time would be two seconds or less. At least 90% of the transactions would be completed in first period, with a response time of less than 0.50 seconds for trivial transactions. Lastly, the total amount of physical paging would be monitored. In this TSO context, physical paging is the total page and swap rate to and from DASD. Therefore, an artificial limit of approximately 400 pages was set as the paging

threshold. Once the threshold was reached, this would define a memory constraint and some action would have to be taken to satisfy the constraint. As the processor becomes faster, this artificial limit may become meaningless.

If the transaction's working set (occupied pages of main storage) could remain in storage at all times, the overall response time would remain both low and consistent. When a transaction or the user went into THINK mode, the transaction's working set could be marked logically swapped out. In the logical swapped state, the transaction remains in storage, but is just marked dormant. If the occupied storage is not required for any other transaction, the transaction stays in storage. However, if the storage becomes constrained, the dormant request can be physically (swapped) removed from main storage and copied to DASD. This of course requires I/O time to move the image from storage to auxiliary storage and additional time to reverse the process. When the user wakes up and enters the next transaction, the copy on auxiliary storage, has to be moved back to main storage. The I/O time is additive and can become a significant part of response time.

The results of the experimental series are documented in the Planning for Capacity - TSO (see [2]) document. The LONG transaction workload was used because of the heavier Problem State versus Supervisor State. In other words, more of the CPU cycles were doing user work instead of supervisor state functions. (See Figure 13.)

Results:

		Data Source
Processor	5860	
Users	170	(TPNS)
Megabytes	48	
CPU Busy	97.81%	(RMF)
TPNS Avg R.T.	1.93 seconds	(TPNS)
TPNS TRX Total	13,715	(TPNS)
Avg. Response Time	2.29 seconds	(RMF)
TRX Total	15,686	(RMF)
1st Period Response Time	.29 seconds	(RMF)
%Trans. in first period	89.91%	(RMF)
Page Rate	422.66/second	(RMF)
ETR	7.62/second	(Calc)

The attained results showed that at 97.81% CPU busy, the total transaction count was 13,715 as reported by TPNS and 15,686 for RMF. The higher transaction count in RMF is attributed to the inclusion of extra transactions. Every time there is an output terminal wait detected because of an output buffer shortage, RMF considers this a transaction. Therefore the transaction count in RMF is actually inflated. TPNS does not consider this a transaction and all of the ETR and ITR calculations in this paper will use the TPNS transaction count as the base. However, TPNS does not give a breakdown of the response times into trivial and other periods. Consequently, both numbers will be reported here and the reader can make independent calculations.

Notice that the physical paging rate (422.66 pages/second) has exceeded the service level agreement to the simulated user. To alleviate the paging constraint, add more storage. This will reduce the amount of CPU cycles required for page movement and more users can be added until the processor is 100% busy. Caution! CPU cycles will be at a premium and response time can suffer.

Also notice that the response time agreements have just about been reached in TPNS (1.93 seconds) and exceeded in RMF (2.29 seconds). If more users were added to the population, physical paging climbed at an alarming rate and as the physically paging requirements went up, so did response time. Again, more storage will bring down the paging rate and more users will be added to take up the available cycles.

Increase the amount of main storage from 48MB to 64MB. (See Figure 14.) Run with an increased number of terminals until the processor is 100% busy. This will probably exceed the service level agreement for average response time, but find out what it takes to max out the processor.

Two hundred users in 64MB will push the processor to 100% busy. (See Figure 15.) Since there are internal queues for the CPU, user response time goes up. First period or trivial response time stays constant, but the jobs that require a lot of CPU time are taking longer to complete. Reason: The processor is saturated, and there are no more cycles available. 64MB over 48MB with this workload gave a 9% increase in throughput, with about a 17% increase in user population. Solution: No more uniprocessor; look at a dual processor. In this environment, there will be two servers working at the same time.

DUAL PROCESSOR

The major difference in this 64MB environment dual processor environment is the two CPUs. (See Figure 16.) Everything else remains constant. The main memory size stayed at 64MB. Since there are two servers, you would expect response time to improve. Can you add more users?

(See Figure 17.) The 5870 dual processor drops in utilization to 78.49%. If more users are added past the 210 range, paging climbs dramatically. There is marked improvement in overall response time (two engines versus one engine). However, the ETR increases by only 1.6 transactions per second. It would appear that the dual processor is a bad decision. Actually, it was a good partial decision. More main storage will alleviate the memory constraints.

The total main storage size is now 96MB. (See Figure 18.) All the rest of the parameters remain constant. The 64MB 5870 experiments could not use up the CPU cycles because an increase in user population will increase paging beyond the agreed limits. In this environment, the number of users will be increased until the maximum CPU busy can be reached. There is a dramatic increase in user population: 65% - 200 to 330!

(See Figure 19.) Notice, with 330 users the processors are backup to 98.20% busy. Response time is up to 1.44 seconds, but it is offset by the 65% increase in user population. The paging rate is well over the 400 pages per second page rate that was agreed upon. In order to cut this down to a more manageable level, it would require more main storage or a less ambitious user population. However, if you cut the user population, the ETR will decrease and so will CPU busy.

This configuration is the maximum amount of main storage that a 5870 can support. (See Figure 20.) The 96MB configuration supported 330 users but ran into a paging rate of 544.62 pages per second. The additional main storage should reduce the paging rate and allow more users into the operational environment. Let's see if it does!

(See Figure 21.) First of all, the page rate dropped by 50%. This void was then filled with at least 20 more users to a 350 maximum. The processor could not or would not go above 98.63% CPU busy. This release of the operating system used the very top 1% for

interprocessor communication. Perhaps it would be beneficial to review the various environments that have been reported on.

(See Figure 22.) The 48MB uniprocessor had a transaction rate of 7.62 transactions per second and the series reached a peak of 16.19 transactions per second in the maximum 128MB configuration. The interesting columns are highlighted to illustrate that going from a uniprocessor to a dual processor without installing additional storage allowed for 10 more users and a Multiprocessing Factor (MP) of 1.20. In other words, the dual processor was providing 20% more throughput than the uniprocessor. (See Figure 23.) When additional main storage was put on the 5870, the user count went from 210 to 330 - the CPU busy went up dramatically. Consequently, the throughput went from 8.29 transactions per second on the uniprocessor to 15.22 transactions per second on the dual processor. This is an implied MP factor of 1.83 with a 65% increase in the number of users. At maximum configuration, the MP factor is 1.95, with a 75% increase in the number of users.

The next step in the progression would be to replace the 5870 with the new 5890-300 and start with 128MB of main storage. There are other models of the 5890 like the 190 and the 200, but let us concentrate on the 300. (See Figure 24)

Questions of interest:

- How many more users can be supported?
- How much faster is the new processor?
- What is the ratio of performance between the two tested processors?
- What happens when you have a new release of the operating system?

FASTER DUAL PROCESSOR

The results of the set of experiments on the new dual processor are reported in the Planning for Capacity - TSO Volume 2 (see [3]). The first column Run #349 was the results of the experiments with MVS 2.1.2. (See Figure 25.) The second column is the results using MVS 2.1.3. All of the new experiments were conducted using 2.1.3; therefore, the base of the 5870 for comparison purposes had to be repeated and Run # 672 will become the new base. Basically, in 128MB of main storage the 550 users produce 23.80 transactions per second in spite of the 2293.00 pages per second of physical paging. The processors are only 74.41% busy, but adding more users would increase the paging rate and elongate the response time. It is basically a repeat of earlier runs; more main storage would be required to bring down the page rate and improve response time. Therefore, in 128MB the 5890-300 processed 52% more transactions than the 5870 while supporting 57% more users.

The main storage area has been increased to 192MB. (See Figure 26.) More of the transactions will remain logically swapped and stay in main storage. In addition more users can be added to the population and the CPU busy metric will go up. Response time will remain constant. But the overall throughput or number of transactions per second will go up.

(See Figure 27.) As anticipated the page rate decreases by approximately 50%. Another 100 users are added to the population. There is also an increase of approximately five more transactions per second in throughput from the 128MB experiments. In retrospect, the 192MB configuration is processing 82% more transactions per second than the 5870, and is supporting 86% more users at the same time. Again, the 1308.58 Page Rate is constraining the system, which only reaches 86.74% CPU busy.

A 256MB main storage size on the 5890-300 is a maximum configuration point. (See Figure 28.) The last environment (192MB) did not use up all of the CPU cycles. Therefore, how many more users can be added to the population? Will logical swapping continue to be beneficial? Can the page rate be reduced even further?

The primary metric shows that the CPU is busy 98.49% of the time and that the user population has grown to 700. (See Figure 29.) The physical paging rate is down to 731.34 pages per second. When you compare the 5890-300 (256MB) with the 5870 (128MB) you can see that the 5890 is processing 94% more transactions with a user population that is 100% greater. All of the experiments to this point have been conducted in main storage configurations.

CONCLUSIONS

If you look at a graphic presentation of the data representing Response Time versus Number of Users, the differences in the various configurations become apparent. (See Figure 30.) Notice that at 300 users the various configurations are not that far apart. However, as the user population increases, the differences become more significant. What two comparisons become obvious?

The first indicator is a performance difference. (See Figure 31.) The 350 user population gives an average response time of 2.30 seconds. This is 0.30 seconds over the agreed service level agreement. The same user population on the 5890-300 gives an average response time of 0.44 seconds. What about a Capacity Difference?

The Capacity Difference can be illustrated by drawing a horizontal line from the 350 user point on the 5870 line and letting it intersect the 5890-300 line. (See Figure 32.) That intersection is at about 680 users. In other words, the 2.30 second response time on the 5870 can be duplicated on the 5890-300 by adding 330 more users.

Next, what is the effect on the capacity difference when main storage is reduced? (See Figure 33.) Draw a horizontal line starting at the 2.00 second response time point. The line will intersect the 128MB representation at approximately 540 users. In 192MB the intersection represents approximately 620 users. Lastly, the intersection of the 256MB line represents approximately 670 users.

At the end of the experimental cycle, someone asked the question, "How many users doing program development work would be required to max out a 5890-300 that had a 256MB main storage?" (See Figure 34.) Using the short transaction workload, the answer was 1020. Also notice that the number of transactions handled was about 118,000. The interesting point was that the RMF report could not report on a user population that was made up of a value that had more than three characters.

If you were to paraphrase what the entertainer says, TSO would like to thank you for the large main memories. (See Figure 35.) Large main storage sizes can accommodate logical swapping options. Remember that large main storage sizes will have to be tuned differently than the real and ESE combinations. The performance analyst will have to be able to sing different tunes. Be aware of the tuning parameters for THINK TIME.

REFERENCES

[1] Lichtig, B.A., "Simulation is 'Unreal'", Symposium on the the Simulation of Computer Systems, SIGSIM, ACM, Boulder, Colorado, August 12-14, 1975, pp. 213-219.

[2] Lichtig, B.A., Martindale, R.W., "Planning for Capacity - TSO", Amdahl Form Number M1195, dated July, 1985.

[3] Lichtig, B.A., Martindale, R.W., Parmelee, T.L., "Planning for Capacity - TSO Volume 2", Amdahl Form Number MM001304, dated December 1986.

[4] Robinson, J.E., Jose, R., "Planning for Capacity - IMS", Amdahl Form Number M1197, dated July 1985.

[5] Robinson, J.E., Young, D.J., "Planning for Capacity - IMS Volume 2", Amdahl Form Number MM001305, dated November 1986.

[6] Robinson, J.E., "Planning for Capacity - CICS," Amdahl Form Number M1196, dated July 1986.

[7] Lockey, M.A., Martindale, R.W., "Planning for Capacity - CICS Volume 2", Amdahl Form Number MM001312.

[8] Lockey, M.A., Jose, R., "Planning for Capacity - VM/CMS", Amdahl Form Number MM001255, dated November 1985.

[9] Major, J.B., "A Methodology for Processor Capacity Comparisons", Proceedings CMG'86, December 1986, pp. 241-252.

P.S.*

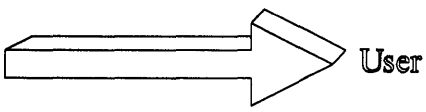
I ♥ YOU

Signed TSO

***P.S. = Processor Storage**

Figure 1

Capacity Planning



?

Figure 2

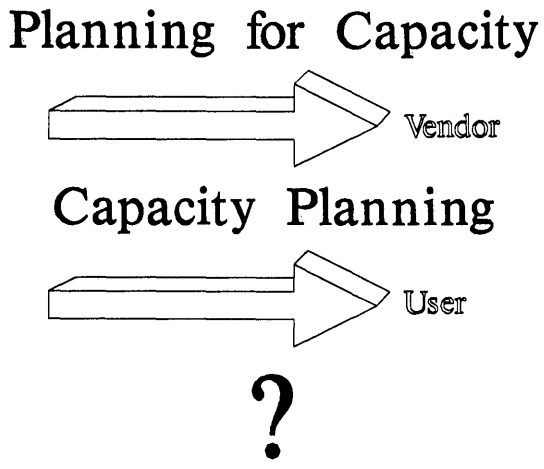


Figure 3

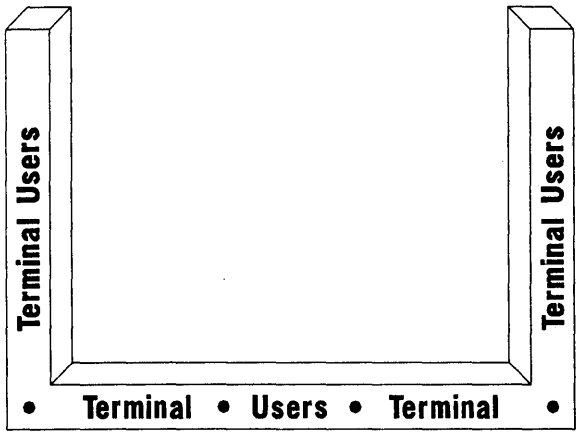


Figure 4

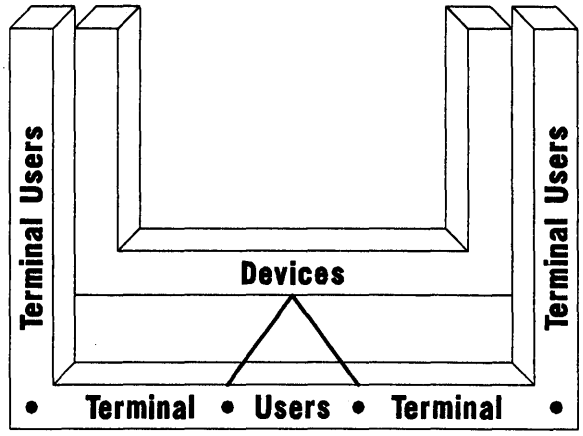


Figure 5

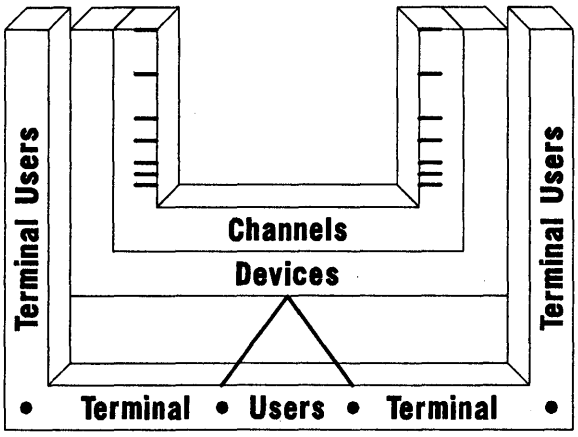


Figure 6

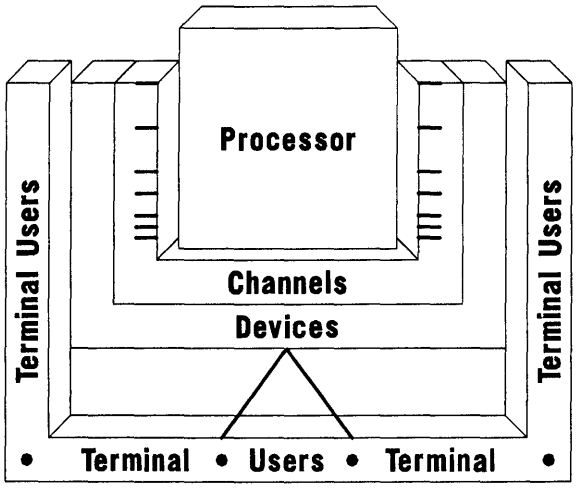
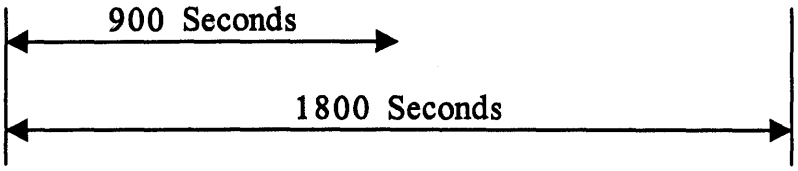


Figure 7

Interval = 1800 Seconds
CPU Busy = 50%



$$\text{External} = \frac{\#\text{ of Transactions}}{\text{Interval Seconds}}$$

$$\text{Internal} = \frac{\#\text{ of Transactions}}{\text{CPU Busy Seconds}}$$

$\therefore$ If CPU Busy = 100% External = Internal

Figure 8

Consider This Example

Run Number	315	346
Data Table	5.2.6	5.2.8
Processor(s)	5870	5870
CPU Busy	61.51%	98.20%
TPNS Trans.	14524	27392
ITR	13.14	15.50
ETR	8.08	15.22

Figure 9

	Without BR 15 (S)	
Processor(s)	5870	5860
CPU Busy	32.89%	39.74%
CPU Busy Secs	296.01	357.66
TPNS Trans.	8878	8879
Internal Transaction Rate	29.99	24.92
MP Factor 5870/5860	1.21	

Figure 10

	With BR 15 (S)		Without BR 15 (S)	
Processor(s)	5870	5860	5870	5860
CPU Busy	100.00%	100.00%	32.89%	39.74%
CPU Busy Secs	157.50	269.00	296.01	357.66
TPNS Trans.	8872	8869	8878	8879
Internal Transaction Rate	56.33	32.97	29.99	24.92
MP Factor 5870/5860	1.71		1.21	

Figure 11

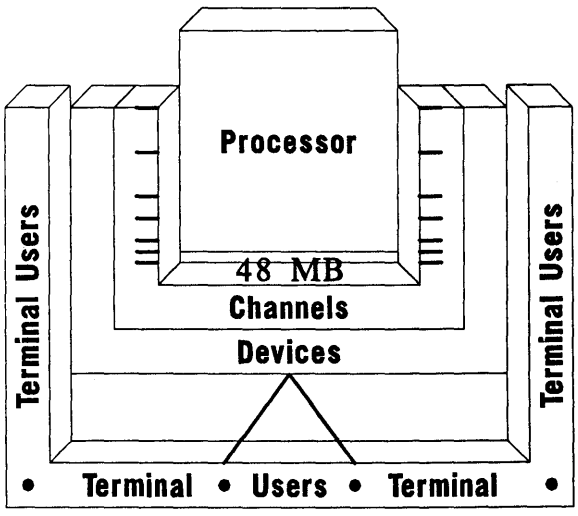


Figure 12

Run#	330
Processor	5860
Users	170
Megabytes	48
CPU Busy	97.81%
TPNS Avg R.T.	1.93
TPNS TRX Total	13715
Avg R.T.	2.29
TRX Total	15686
1st per R.T.	.29
% TRX/1st	89.91
Page Rate	422.66
External TRX Rate	7.62

Figure 13

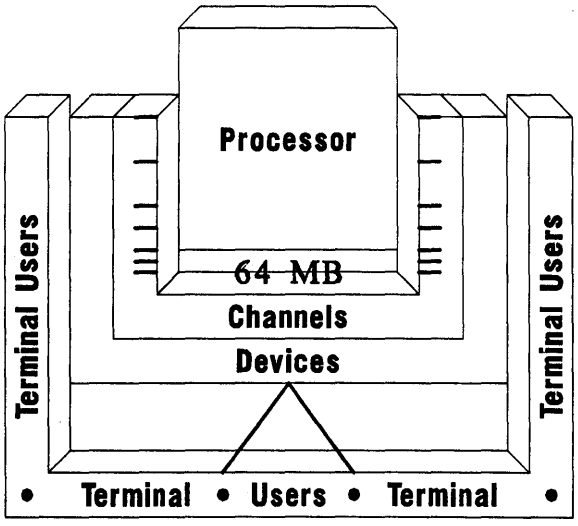


Figure 14

Run#	330	350
Processor	5860	5860
Users	170	200
Megabytes	48	64
CPU Busy	97.81%	99.96%
TPNS Avg R.T.	1.93	3.72
TPNS TRX Total	13715	14916
Avg R.T.	2.29	3.98
TRX Total	15686	17121
1st per R.T.	.29	.26
% TRX/1st	89.91	90.22
Page Rate	422.66	312.81
External TRX Rate	7.62	8.29

Figure 15

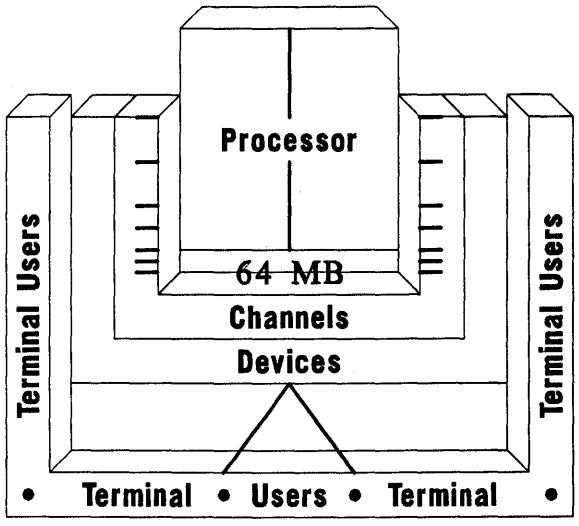


Figure 16

Run#	330	350	318
Processor	5860	5860	5870
Users	170	200	210
Megabytes	48	64	64
CPU Busy	97.81%	99.96%	78.49%
TPNS Avg R.T.	1.93	3.72	.73
TPNS TRX Total	13715	14916	17957
Avg R.T.	2.29	3.98	.86
TRX Total	15686	17121	20763
1st per R.T.	.29	.26	.20
% TRX/1st	89.91	90.22	89.87
Page Rate	422.66	312.81	350.95
External TRX Rate	7.62	8.29	9.87

Figure 17

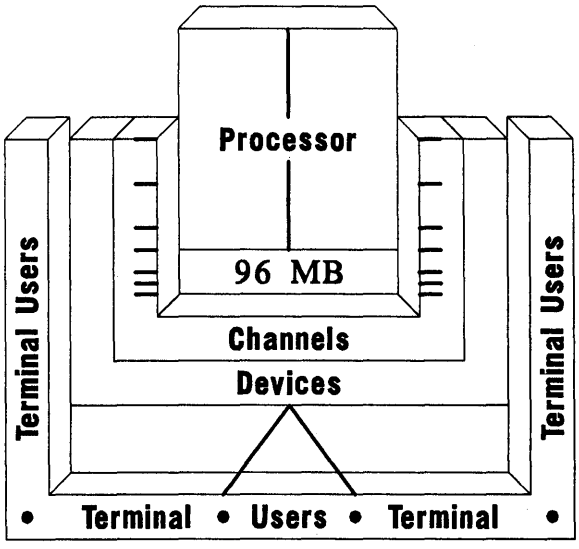


Figure 18

Run#	346
Processor	5870
Users	330
Megabytes	96
CPU Busy	98.20%
TPNS Avg R.T.	1.44
TPNS TRX Total	27392
Avg R.T.	1.67
TRX Total	31746
1st per R.T.	.30
% TRX/1st	89.59
Page Rate	544.62
External TRX Rate	15.22

Figure 19

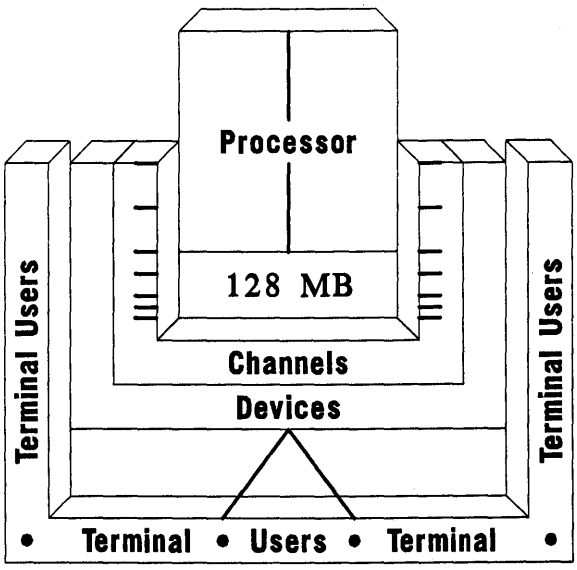


Figure 20

Run#	672	704	695
Processor	5870	5890	5890
Users	350	550	650
Megabytes	128	128	192
CPU Busy	97.92	74.41	86.74
TPNS Avg R.T.	2.30	2.96	2.66
TPNS TRX Total	28125	42841	51261
Avg R.T.	1.70	2.19	2.04
TRX Total	46770	65428	80601
1st per R.T.	.17	1.68	1.40
% TRX/1st	91.72	91.75	91.44
Page Rate	348.05	2293.00	1308.58
External TRX Rate	15.63	23.80	28.48

Configuration		Ratios	5890/5870
5890-300	5870	Transactions/sec	Users
192 MB	128 MB	1.82	1.86
128 MB	128 MB	1.52	1.57

Figure 21

Run#	330	350	318	346	349
Processor	5860	5860	5870	5870	5870
Users	170	200	210	330	350
Megabytes	48	64	64	96	128
CPU Busy	97.81%	99.96%	78.49%	98.20%	98.63%
TPNS Avg R.T.	1.93	3.72	.76	1.44	1.35
TPNS TRX Total	13715	14916	17957	27392	29135
Avg R.T.	2.29	3.98	.86	1.67	1.61
TRX Total	15686	17121	20763	31746	34093
1st per R.T.	.29	.26	.20	.30	.21
% TRX/1st	89.91	90.22	89.87	89.59	88.58
Page Rate	422.66	312.81	350.95	544.62	267.92
External TRX Rate	7.62	8.29	9.87	15.22	16.19

Configuration		MP Ratios	
5870	5860	Transactions/sec	Users
128 MB	64 MB	1.95	1.75
96 MB	64 MB	1.83	1.65
64 MB	64 MB	1.20	1.05
128 MB	48 MB	2.12	2.05
96 MB	48 MB	2.00	1.94

Figure 22

Run#	330	350	318	346	349
Processor	5860	5860	5870	5870	5870
Users	170	200	210	330	350
Megabytes	48	64	64	96	128
CPU Busy	97.81%	99.96%	78.49%	98.20%	98.63%
TPNS Avg R.T.	1.93	3.72	.76	1.44	1.35
TPNS TRX Total	13715	14916	17957	27392	29135
Avg R.T.	2.29	3.98	.86	1.67	1.61
TRX Total	15686	17121	20763	31746	34093
1st per R.T.	.29	.26	.20	.30	.21
% TRX/1st	89.91	90.22	89.87	89.59	88.58
Page Rate	422.66	312.81	350.95	544.62	267.92
External TRX Rate	7.62	8.29	9.87	15.22	16.19

Configuration		MP Ratios	
5870	5860	Transactions/sec	Users
128 MB	64 MB	1.95	1.75
96 MB	64 MB	1.83	1.65
64 MB	64 MB	1.20	1.05
128 MB	48 MB	2.12	2.05
96 MB	48 MB	2.00	1.94

Figure 23

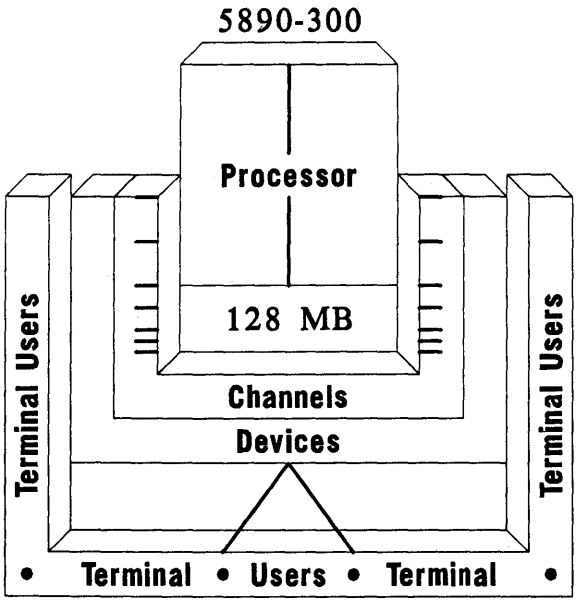


Figure 24

Run#	349	672	704
Processor	5870	5870	5890
Users	350	350	550
Megabytes	128	128	128
CPU Busy	98.63%	97.92	74.41
TPNS Avg R.T.	1.35	2.30	2.96
TPNS TRX Total	29135	28125	42841
Avg R.T.	1.61	1.70	2.19
TRX Total	34093	46770	65428
1st per R.T.	.21	.17	1.68
% TRX/1st	88.58	91.72	91.75
Page Rate	267.97	348.05	2293.00
External TRX Rate	16.19	15.63	23.80

Configuration		Ratios	5890/5870
5890-300	5870	Transactions/sec	Users
128 MB	128 MB	1.52	1.57

Figure 25

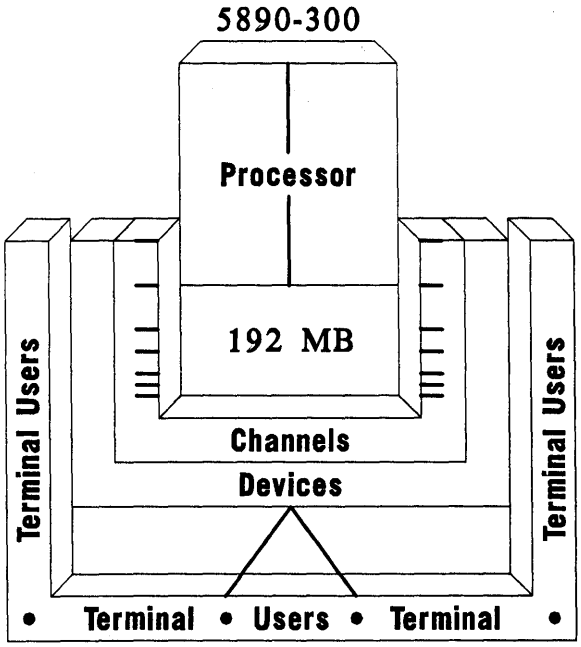


Figure 26

Run#	672	704	695
Processor	5870	5890	5890
Users	350	550	650
Megabytes	128	128	192
CPU Busy	97.92	74.41	86.74
TPNS Avg R.T.	2.30	2.96	2.66
TPNS TRX Total	28125	42841	51261
Avg R.T.	1.70	2.19	2.04
TRX Total	46770	65428	80601
1st per R.T.	.17	1.68	1.40
% TRX/1st	91.72	91.75	91.44
Page Rate	348.05	2293.00	1308.58
External TRX Rate	15.63	23.80	28.48

Configuration		Ratios	5890/5870
5890-300	5870	Transactions/sec	Users
192 MB	128 MB	1.82	1.86
128 MB	128 MB	1.52	1.57

Figure 27

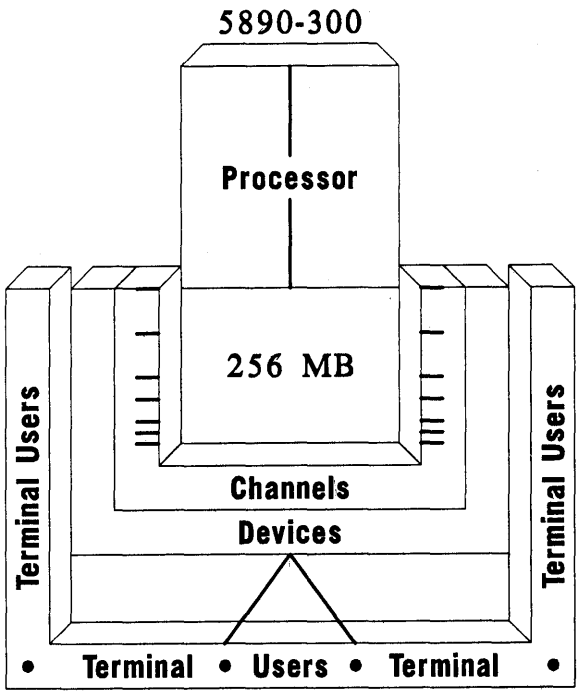


Figure 28

Run#	672	704	695	683
Processor	5870	5890	5890	5890
Users	350	550	650	700
Megabytes	128	128	192	256
CPU Busy	97.92	74.41	86.74	98.49
TPNS Avg R.T.	2.30	2.96	2.66	2.92
TPNS TRX Total	28125	42841	51261	54534
Avg R.T.	1.70	2.19	2.04	2.04
TRX Total	46770	65428	80601	85275
1st per R.T.	.17	1.68	1.40	.87
% TRX/1st	91.72	91.75	91.44	90.97
Page Rate	348.05	2293.00	1308.58	731.34
External TRX Rate	15.63	23.80	28.48	30.30

Configuration		Ratios	5890/5870
5890-300	5870	Transactions/sec	Users
256 MB	128 MB	1.94	2.00
192 MB	128 MB	1.82	1.86
128 MB	128 MB	1.52	1.57

Figure 29

Average Response Time (sec) vs Number of Users

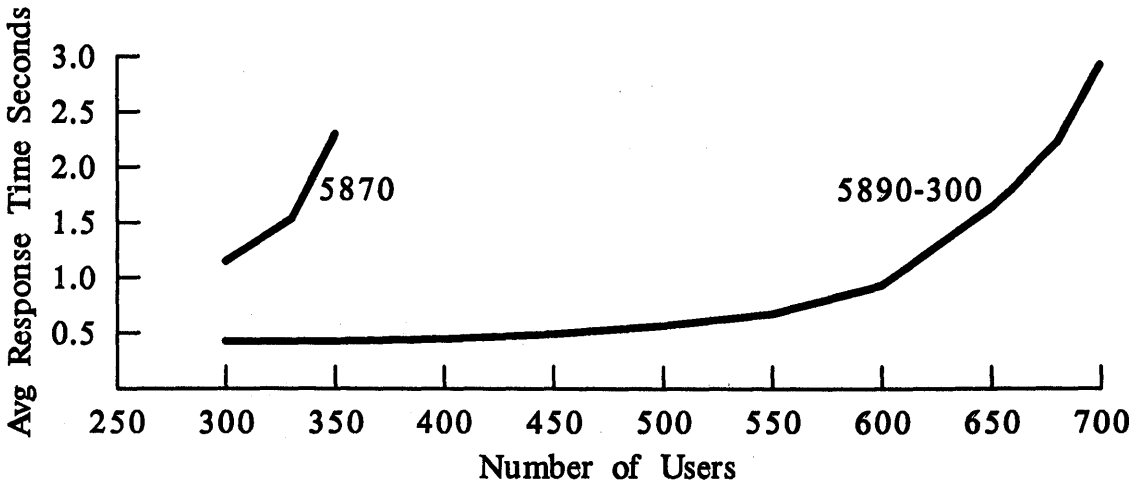


Figure 30

Average Response Time (sec) vs Number of Users

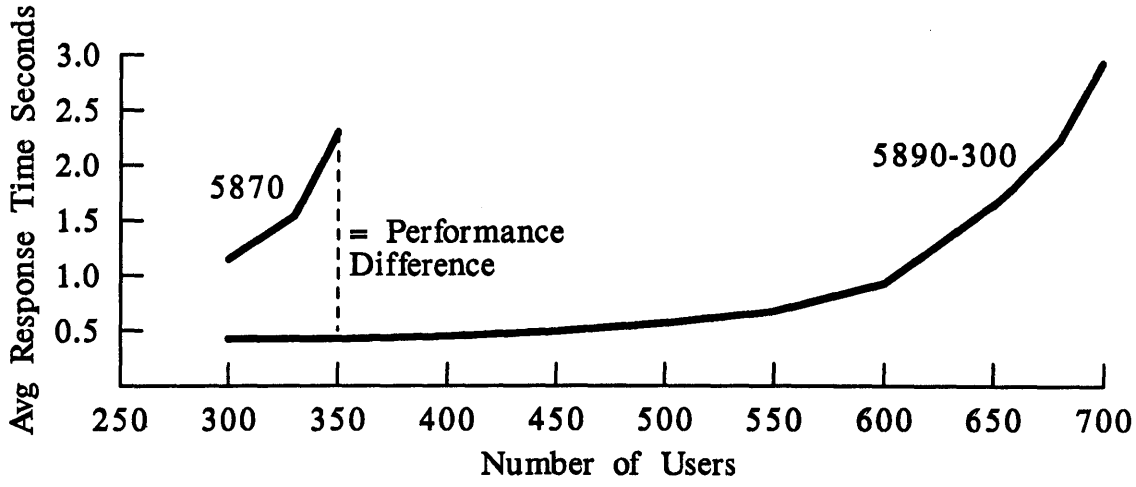


Figure 31

Average Response Time (sec) vs Number of Users

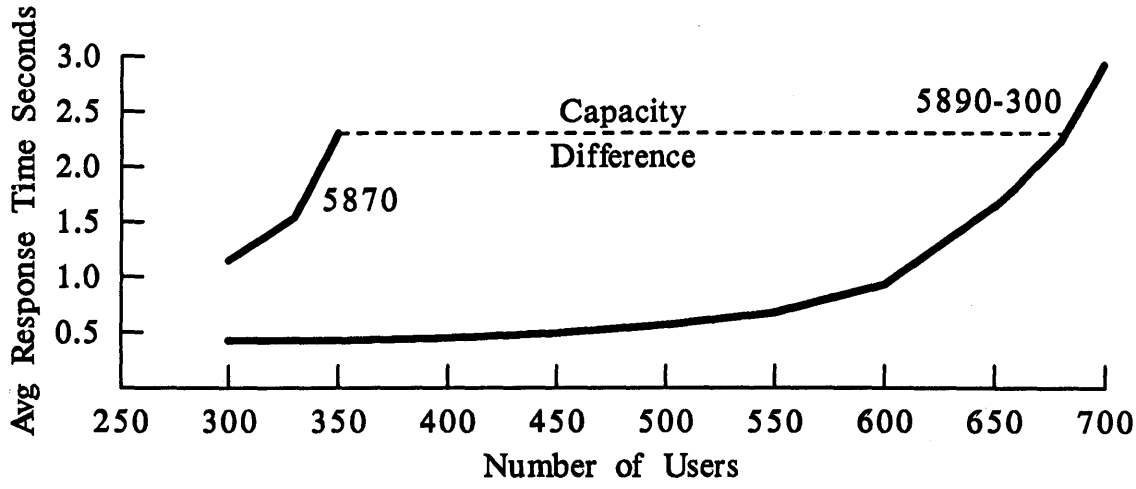


Figure 32

Average Response Time (sec) vs Number of Users

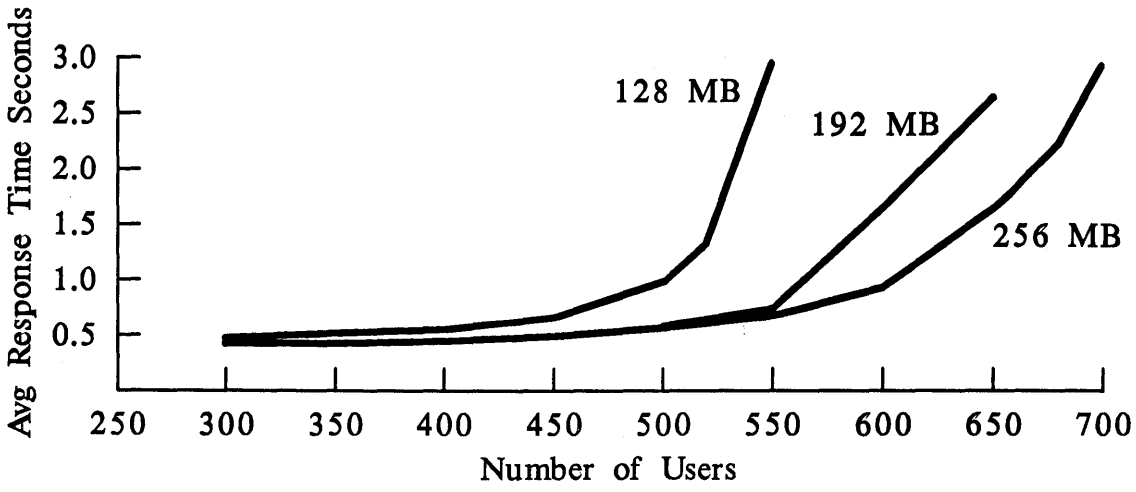


Figure 33

Run#	729
Processor	5890-300
Users	1020
Megabytes	256
CPU Busy	96.14%
TRX Total	118,848
Avg R.T.	2.373
1st per R.T.	.481
% TRX/1st	94.25
Page Rate	1585.28
External TRX Rate	66.03

Figure 34

Figure 35

A CAPACITY PLANNING METHODOLOGY FOR DATA NETWORKS

John Harmon
Lumbermens Mutual Casualty Company

ABSTRACT

Planning capacity requirements for data networks can be a very difficult task. Not only do we deal with several interrelated resources, but we must also work with potentially awkward management structures that have different units responsible for different parts of the network.

This paper presents a hierarchical structure for projecting usage requirements on each of the components in a typical multidomain leased line network. It deals with providing a technique for characterizing data in terms of differing critical time frames for each component; classifying the individual components; defining the relationships between each of the component classes; and showing how this structure can ease the planning burden of those responsible for the individual pieces of the network.

INTRODUCTION

Maximizing an organization's data communications investment involves an organized capacity planning approach. Unfortunately, while mainframe capacity planning seems to have achieved an aura of respectability, network capacity planning is something of an orphan. In fact, the meaning of the phrase "network capacity planning" can have widely different meanings depending on the organization.

For the purposes of this paper, the realm of network capacity planning will be defined to include capacity planning for both the communications lines in our sample network and capacity planning for 3725 Communications Controllers in the network. The paper describes a comprehensive capacity planning sequence in which devices attached to the data network have, as a part of the capacity plan, a description of the data flow going to and from the network. The purpose of this sequence is to improve and coordinate each of several capacity plans.

For the sake of simplification, All examples will refer to an SNA multidomain network where all of the equipment is IBM, and where each 3725 is located at the same datacenter as a host mainframe. This is not to say that this is the only situation where this technique will work, but that it will simplify the explanation. A diagram of this network is in the Appendix.

THE PROBLEM

There are several reasons that network capacity has not received the attention it deserves:

(1) First, network capacity planning involves several different kinds of devices in several different locations. Support personnel knowledgeable in all of the components are spread throughout the company and may seldom get an opportunity to interact.

(2) Responsibility for the capacity of each of the components of a network can be spread across political boundaries within a company. The remote office or an application development team could be responsible for a System 36 at the remote office, the telecommunications department is responsible for the telephone lines and modems, the network software support group is responsible for the communications controller (3725), and the Systems Software unit is responsible for the host mainframe. Even when these groups interact cooperatively, they often have hidden agendas which have the potential to cause problems. A simple example might be a situation where the telecommunications department wants to save the costs of some lines by purchasing some higher speed modems. After the changes are made, the network support group may find that the 3725 is getting errors and belatedly discover that the 3725 can't support the new linespeeds without being reorganized.

(3) Time frames for doing capacity planning on the individual pieces of the network are done at different times of the year. In most cases, mainframe capacity planning and telecommunications capacity planning are geared to complete as the budget cycle begins. Unfortunately, much of what the telecommunications people may need to do adequate planning will not become available until after their capacity plan is done.

Similarly, the network support unit may actually plan 3725 usage, but the Network Control Center may pay for it. In order to do a good job, the network support unit will probably want to use data made available when mainframe and telecommunications planning is done. Their results can be quite good, but anything they produce may be too late for the NCC manager to put in his budget projections.

If the end user does his own planning, as is often the case when all he has is control units and terminals, he, too, will time his requests to feed into the budget cycle, without considering potential impact on the telephone lines. Conversely, if the end user has a minicomputer such as System 36 or an 8100, capacity planning is usually done by either an application support group or a device support group. In either case, capacity planning will probably be done without regard to either telephone line needs or budget cycle timing.

What is needed, then, is an organizational method for performing network capacity planning so that:

(1) There is a uniformity of timing of capacity planning for devices of similar function. These will be defined later.

(2) Capacity Planning information flow must be prearranged so that the input and output requirements for each device are known and agreed to before any capacity planning begins.

(3) All levels of capacity planning must be completed in time to feed into the budget cycle.

ORGANIZATION OF NETWORK LEVELS

The first step of this organizational method is to define the levels of network components. This would be done as follows:

(1) Level One - These devices include end user devices such as control units (3274's), end user minicomputers (System 36's or 8100's), and host mainframes. The common thread is that capacity

planning of each of these devices has as one of its prime inputs, a user estimate of some business activity. While much of what is determined in the device capacity plans feeds into the mainframe capacity plan, as far as the network is concerned, they must be considered as similar devices.

(2) Level Two - These are the telephone lines, but only those which connect Level One devices with 3725's. Input to this level is output from Level One.

(3) Level Three - These are the lines connecting the 3725's in the network. As mentioned before, this assumes that each 3725 in the network is channel-attached to a host mainframe. Input to this level comes from Level One.

(4) Level Four - These are the 3725's in the network. Input to this level comes from Level Two and Level Three.

WHAT INFORMATION GETS TRANSFERRED BETWEEN LEVELS

This approach assumes that at each level, some sort of capacity planning is being done for each and every device. What the method demands is that the capacity plan for each and every device include a document detailing data flow such that it can be used at the succeeding levels. Note that it will save a good deal of trouble if each and every device in the network is given a unique name, preferably with some indication as to what kind of device it is. Keep in mind, however, at some point, someone is going to add (or delete) a component (device, line, or even a 3725). This is fine, just as long as people know how to apply the naming convention.

Input to Level One is the basic user business projection. The driving force for all levels of capacity planning is, of course, how much work is expected to be done. The "hows" of capacity planning at each of these levels will, of course, depend upon cost benefit analyses by whoever is responsible, but, minimally, it will include an output document with certain distinct pieces of data. These are:

(1) The name of the device.

(2) The name of the 3725 to which the device is attached.

(3) The name of the mainframe host which owns it (may be the same as number one).

(4) The name of host on the network to which this traffic estimate applies (need not be the same as number three).

(5) The name of the 3725 to which the host device is attached.

(6) The highest number of characters per second that will be sent to the host named in number four above for each hour throughout the day.

(7) The highest number of characters per second that will be received from the host named in number four above for each hour throughout the day.

(8) For interactive device controllers, the number of transactions initiated by terminals at the remote device.

(9) For batch device controllers, the number of blocks of data sent to or from the remote device.

(10) The number of subsidiary devices (such as terminals) which must be known by the network as associated with number one.

An example of this document is shown in Figure 1.

Note that if this is done for each of 200 devices on a network that contains five host mainframes, it will involve a great deal of paper. However, it lends itself to automation very easily.

Network input to Level Two consists of a simple summation of the output records from Level One by device, by 3725. Whoever does the network design for the boundary networks (remote office to 3725) can then use this information to perform capacity planning. Output from this level also includes a form for each line specifying the following information:

(1) The name of the line.

(2) The name of the 3725 to which the line is attached.

(3) The speed of the line (essentially, the number of characters per second that the line can carry).

(4) The number of primary devices.

(5) The number of subsidiary devices (terminals).

(6) The highest number of characters per second that will be sent to the devices on this line for each hour throughout the day.

(7) The highest number of characters per second that will be received by the 3725 on this line for each hour in the day.

An example of this document is shown in Figure 2.

Network input to Level Three consists of a summary of those input records from Level One in which the attached 3725 and the destination 3725 differ. This indicates that they need to be connected by an internetwork node link (3725 to 3725) rather than by a boundary link. Notice that this will include host to host traffic such as dataset transmissions. This information will probably be given to someone in the telecommunications and/or network support unit to determine what circuits will be necessary to support this traffic.

It may seem that Level Three is similar enough to Level Two to combine the two steps. Resist this approach. First of all, planning Level Three links (INN links) is more complex than planning Level Two links, so you would want to keep the distinction between the two levels for as long as possible. Secondly, the 3725 (which is configured in Level Four) handles boundary links (Level Two) differently than it handles INN links (Level Three). Thirdly, many companies which have only one datacenter will not have to worry about Level Three at all. To try to include Level Three information where people are not familiar with it might introduce a confusion which is not necessary.

In addition to putting out changes in the backbone (INN) network, whoever plans at this level will produce a document for each INN link indicating the following information:

(1) Name of the link.

(2) Speed of the link.

(3) Name of the "A" side 3725 ("A" side is arbitrary).

(4) Name of the host mainframe that owns the 3725 in number two.

(5) Name of the "B" side 3725 ("B" side is arbitrary).

(6) Name of the host mainframe that owns the 3725 in number four.

(7) The highest number of characters per second that will be sent from the "A" 3725 to the "B" 3725 by each hour of the day.

(8) The highest number of characters per second that will be sent from the "B" 3725 to the "A" 3725 by each hour of the day.

An example of this document is shown in Figure 3.

Network input to Level Four consists of the documents produced by both Levels Two and Three. What is essentially provided is data that can be passed to the proper person, probably in the network support area, who will plan for the capacity of the 3725's. There will be no output documents from Level Four, other than the 3725 capacity plan.

Figure 4 shows each of the steps in the network capacity planning process, along with the associated outputs.

It is important to keep in mind that the suggested data flow documents may be inappropriate for your network. However, they can serve as a starting point for discussions with the people who have to determine exactly what it is that they do need.

MAKING IT WORK

This methodology will not, in and of itself, solve all of your network capacity problems. The key to making it work is to have an informed person serve as an overall Project Manager. The primary roles of the Project Manager are:

(1) Identify and name all of the components (devices, lines, 3725's) in the network. The Network Support Unit can provide much of this information with the NCP source code.

(2) Categorize these devices as to where they fall in the levels described above.

(3) Determine who is going to have primary responsibility for the capacity plan for each network component.

(4) Interview each of those people and determine how long it will take them to (a) complete all of their capacity plans and (b) complete their specific Network Data Flow documents. In most cases, the Data Flow document can be produced first.

(5) Lay out the time line for the Project so that you give each of the planners sufficient time to produce their Data Flow documents and so that all capacity plans are complete in time to serve as input to the budget process.

(6) Monitor the process and make sure that people and technical deficiencies are dealt with as quickly as possible.

I would anticipate that, as with any new method, there would be opposition to this procedure. People planning for the remote devices may have new demands put on them. People doing mainframe planning may not appreciate the imposition of a new schedule. Nevertheless, there are some major paybacks to the effective use of this process:

(1) The timing of the process enables more complete budgeting for network costs.

(2) The flow of the process allows for capacity plans that are more accurate, and more internally consistent.

(3) The preplanning involved makes it less likely that one or more components will fall through the cracks. Because of the hierarchical nature of this method all of the parts must be looked at. This may generate some surprises.

(4) The existence of such a process will help justify the tools necessary to do adequate planning.

(5) Centralized management of the process will assure continuity, standards, and timing of the production of all of the capacity plans which are connected with network planning.

CONCLUSION

In order to provide for adequate input into a network capacity plan, those people who provide the necessary information for each of the parts must be coordinated if the final product is to reach its maximum potential as a management tool. This paper has outlined a way that I believe this coordination can be performed.

If an organization has the basic tools needed for capacity planning and network design; if it has a management desire for and commitment to improved resource planning; and if it has a person with the skills to "ride herd" on the process; the result should be not only improved planning for the telephone lines and communications controllers, but increased effectiveness of resource planning throughout the company.

LEVEL ONE (END USER) NETWORK
CAPACITY PLANNING DOCUMENT

****
* This document should be filled out for each *
* device (control unit or minicomputer) that is *
* in the network. *
****

PLANNER_____ DEPT_____ DATE_____

DEVICE NAME:_____ NO. OF TERMINALS_____

OWNING 3725 NAME:_____ OWNING HOST NAME:_____

DEST HOST NAME:_____ DEST 3725 NAME:_____

DATA SENT TO DEVICE
Characters Per Second

Time		Time	
2400-0059		1200-1259	
0100-0159		1300-1359	
0200-0259		1400-1459	
0300-0359		1500-1559	
0400-0459		1600-1659	
0500-0559		1700-1759	
0600-0659		1800-1859	
0700-0759		1900-1959	
0800-0859		2000-2059	
0900-0959		2100-2159	
1000-1059		2200-2259	
1100-1159		2300-2359	

DATA SENT FROM DEVICE
Characters Per Second

Time		Time	
2400-0059		1200-1259	
0100-0159		1300-1359	
0200-0259		1400-1459	
0300-0359		1500-1559	
0400-0459		1600-1659	
0500-0559		1700-1759	
0600-0659		1800-1859	
0700-0759		1900-1959	
0800-0859		2000-2059	
0900-0959		2100-2159	
1000-1059		2200-2259	
1100-1159		2300-2359	

****

_ TRANSACTIONS SENT TO DEVICE
_ BLOCKS
Transactions Per Second or
Blocks Per Second

Time		Time	
2400-0059		1200-1259	
0100-0159		1300-1359	
0200-0259		1400-1459	
0300-0359		1500-1559	
0400-0459		1600-1659	
0500-0559		1700-1759	
0600-0659		1800-1859	
0700-0759		1900-1959	
0800-0859		2000-2059	
0900-0959		2100-2159	
1000-1059		2200-2259	
1100-1159		2300-2359	

_ TRANSACTIONS SENT FROM DEVIC
_ BLOCKS
Transactions Per Second or
Blocks Per Second

Time		Time	
2400-0059		1200-1259	
0100-0159		1300-1359	
0200-0259		1400-1459	
0300-0359		1500-1559	
0400-0459		1600-1659	
0500-0559		1700-1759	
0600-0659		1800-1859	
0700-0759		1900-1959	
0800-0859		2000-2059	
0900-0959		2100-2159	
1000-1059		2200-2259	
1100-1159		2300-2359	

Figure 1.
Level One Data Flow

LEVEL TWO (BOUNDARY LINK) NETWORK
CAPACITY PLANNING DOCUMENT

****
* This document should be filled out for each *
* line which connects a 3725 and a remote device*
****

NAME OF PLANNER _____

DEPARTMENT _____ DATE _____

****

LINE NAME: _____

OF DEVICES:
(CU's or Minicomputers) _____

OF TERMINALS: _____

SPEED OF LINE: _____

OWNING 3725 NAME: _____

OWNING HOST NAME: _____

****

DATA SENT TO DEVICE
Characters Per Second

Time		Time	
2400-0059		1200-1259	
0100-0159		1300-1359	
0200-0259		1400-1459	
0300-0359		1500-1559	
0400-0459		1600-1659	
0500-0559		1700-1759	
0600-0659		1800-1859	
0700-0759		1900-1959	
0800-0859		2000-2059	
0900-0959		2100-2159	
1000-1059		2200-2259	
1100-1159		2300-2359	

DATA SENT FROM DEVICE
Characters Per Second

Time		Time	
2400-0059		1200-1259	
0100-0159		1300-1359	
0200-0259		1400-1459	
0300-0359		1500-1559	
0400-0459		1600-1659	
0500-0559		1700-1759	
0600-0659		1800-1859	
0700-0759		1900-1959	
0800-0859		2000-2059	
0900-0959		2100-2159	
1000-1059		2200-2259	
1100-1159		2300-2359	

****

Figure 2.
Level Two Data Flow

LEVEL THREE (INN LINK) NETWORK
CAPACITY PLANNING DOCUMENT

****
* This document should be filled out for each *
* link which connects two 3725's. *
****

NAME OF PLANNER _____

DEPARTMENT _____ DATE _____

****

LINK NAME: _____

SPEED OF LINK: _____

"A" SIDE 3725 NAME: _____

"A" SIDE HOST NAME: _____

"B" SIDE 3725 NAME: _____

"B" SIDE HOST NAME: _____

****

FROM "A" TO "B"
Characters Per Second

Time		Time	
2400-0059		1200-1259	
0100-0159		1300-1359	
0200-0259		1400-1459	
0300-0359		1500-1559	
0400-0459		1600-1659	
0500-0559		1700-1759	
0600-0659		1800-1859	
0700-0759		1900-1959	
0800-0859		2000-2059	
0900-0959		2100-2159	
1000-1059		2200-2259	
1100-1159		2300-2359	

FROM "B" TO "A"
Characters Per Second

Time		Time	
2400-0059		1200-1259	
0100-0159		1300-1359	
0200-0259		1400-1459	
0300-0359		1500-1559	
0400-0459		1600-1659	
0500-0559		1700-1759	
0600-0659		1800-1859	
0700-0759		1900-1959	
0800-0859		2000-2059	
0900-0959		2100-2159	
1000-1059		2200-2259	
1100-1159		2300-2359	

****

Figure 3.
Level Three Data Flow

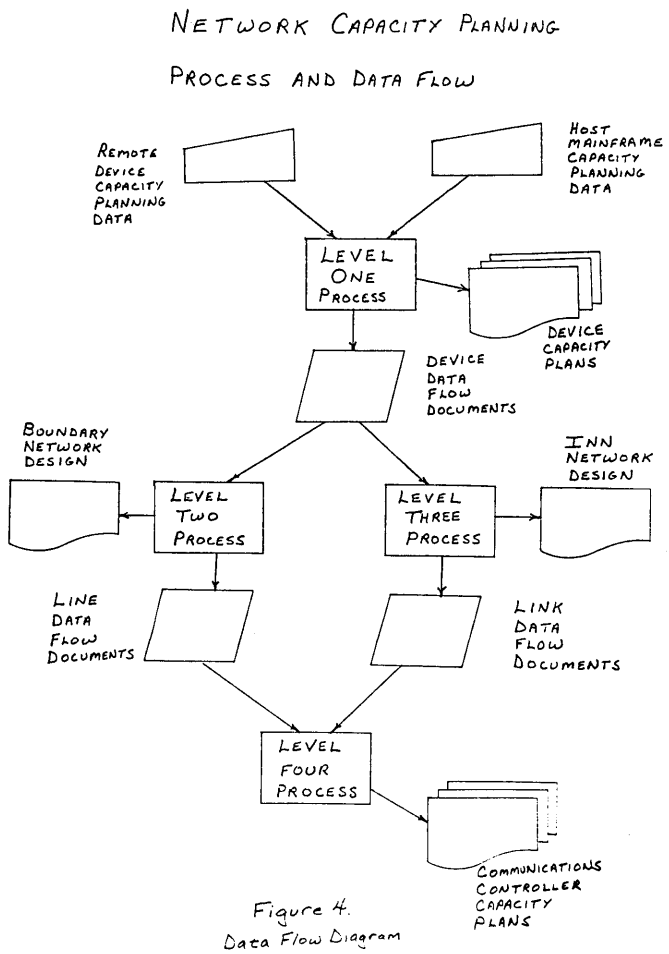
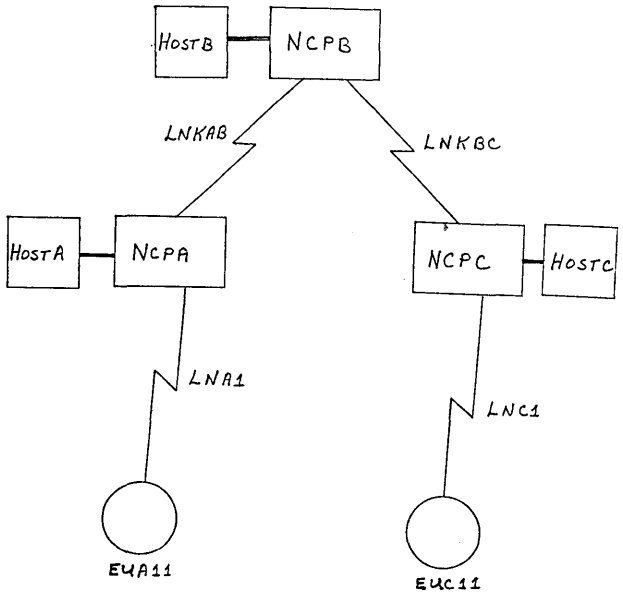


Figure 4.
Data Flow Diagram

APPENDIX

Below and on the following pages are the diagram of a sample network along with sample output documents at each stage of the planning process. The network has been simplified so that the relationships between the documents from each level can be seen.

SAMPLE NETWORK LAYOUT



LEVEL ONE (END USER) NETWORK
CAPACITY PLANNING DOCUMENT

* This document should be filled out for each device (control unit or minicomputer) that is in the network.

PLANNER JAMES BROWN DEPT APPL SUPPORT DATE 9/1/87

DEVICE NAME: EUA11 NO. OF TERMINALS 27

OWNING 3725 NAME: NCPA OWNING HOST NAME: HOSTA

DEST HOST NAME: HOSTB DEST 3725 NAME: NCPB

DATA SENT TO DEVICE
Characters Per Second

Time		Time	
2400-0059	105	1200-1259	150
0100-0159	117	1300-1359	105
0200-0259	133	1400-1459	0
0300-0359	66	1500-1559	0
0400-0459	32	1600-1659	0
0500-0559	0	1700-1759	0
0600-0659	0	1800-1859	0
0700-0759	0	1900-1959	0
0800-0859	0	2000-2059	48
0900-0959	0	2100-2159	106
1000-1059	0	2200-2259	138
1100-1159	0	2300-2359	54

DATA SENT FROM DEVICE
Characters Per Second

Time		Time	
2400-0059	17	1200-1259	27
0100-0159	18	1300-1359	16
0200-0259	22	1400-1459	0
0300-0359	10	1500-1559	0
0400-0459	5	1600-1659	0
0500-0559	0	1700-1759	0
0600-0659	0	1800-1859	0
0700-0759	0	1900-1959	0
0800-0859	0	2000-2059	10
0900-0959	0	2100-2159	17
1000-1059	0	2200-2259	20
1100-1159	0	2300-2359	8

TRANSACTIONS SENT TO DEVICE
~~BLOCKS~~ avg 100 char/trans
Transactions Per Second or Blocks Per Second

Time		Time	
2400-0059	1.05	1200-1259	1.5
0100-0159	1.17	1300-1359	1.05
0200-0259	1.33	1400-1459	0
0300-0359	.66	1500-1559	0
0400-0459	.32	1600-1659	0
0500-0559	0	1700-1759	0
0600-0659	0	1800-1859	0
0700-0759	0	1900-1959	0
0800-0859	0	2000-2059	.48
0900-0959	0	2100-2159	1.06
1000-1059	0	2200-2259	1.38
1100-1159	0	2300-2359	.54

TRANSACTIONS SENT FROM DEVICE
~~BLOCKS~~ avg 20 char/trans
Transactions Per Second or Blocks Per Second

Time		Time	
2400-0059	.85	1200-1259	1.35
0100-0159	.90	1300-1359	.80
0200-0259	1.10	1400-1459	0
0300-0359	.50	1500-1559	0
0400-0459	.25	1600-1659	0
0500-0559	0	1700-1759	0
0600-0659	0	1800-1859	0
0700-0759	0	1900-1959	0
0800-0859	0	2000-2059	.50
0900-0959	0	2100-2159	.85
1000-1059	0	2200-2259	1.00
1100-1159	0	2300-2359	.40

LEVEL ONE (END USER) NETWORK
CAPACITY PLANNING DOCUMENT

* This document should be filled out for each device (control unit or minicomputer) that is in the network.

PLANNER JAMES BROWN DEPT APPL SUPPORT DATE 9/1/87

DEVICE NAME: EUA11 NO. OF TERMINALS 27

OWNING 3725 NAME: NCPA OWNING HOST NAME: HOSTA

DEST HOST NAME: HOSTA DEST 3725 NAME: NCPA

DATA SENT TO DEVICE
Characters Per Second

Time		Time	
2400-0059	0	1200-1259	107
0100-0159	0	1300-1359	382
0200-0259	0	1400-1459	441
0300-0359	0	1500-1559	300
0400-0459	0	1600-1659	126
0500-0559	0	1700-1759	55
0600-0659	0	1800-1859	50
0700-0759	181	1900-1959	0
0800-0859	262	2000-2059	0
0900-0959	337	2100-2159	0
1000-1059	419	2200-2259	0
1100-1159	303	2300-2359	0

DATA SENT FROM DEVICE
Characters Per Second

Time		Time	
2400-0059	0	1200-1259	14
0100-0159	0	1300-1359	58
0200-0259	0	1400-1459	70
0300-0359	0	1500-1559	43
0400-0459	0	1600-1659	16
0500-0559	0	1700-1759	9
0600-0659	0	1800-1859	9
0700-0759	23	1900-1959	0
0800-0859	34	2000-2059	0
0900-0959	46	2100-2159	0
1000-1059	60	2200-2259	0
1100-1159	42	2300-2359	0

TRANSACTIONS SENT TO DEVICE
~~BLOCKS~~ avg 500 char/transaction
Transactions Per Second or Blocks Per Second

Time		Time	
2400-0059	0	1200-1259	.21
0100-0159	0	1300-1359	.76
0200-0259	0	1400-1459	.88
0300-0359	0	1500-1559	.60
0400-0459	0	1600-1659	.25
0500-0559	0	1700-1759	.11
0600-0659	0	1800-1859	.10
0700-0759	.36	1900-1959	0
0800-0859	.52	2000-2059	0
0900-0959	.67	2100-2159	0
1000-1059	.84	2200-2259	0
1100-1159	.61	2300-2359	0

TRANSACTIONS SENT FROM DEVICE
BLOCKS avg 50 char/transaction
Transactions Per Second or Blocks Per Second

Time		Time	
2400-0059	0	1200-1259	.28
0100-0159	0	1300-1359	1.16
0200-0259	0	1400-1459	1.40
0300-0359	0	1500-1559	.86
0400-0459	0	1600-1659	.32
0500-0559	0	1700-1759	.18
0600-0659	0	1800-1859	.18
0700-0759	.46	1900-1959	0
0800-0859	.68	2000-2059	0
0900-0959	.92	2100-2159	0
1000-1059	1.20	2200-2259	0
1100-1159	.84	2300-2359	0

LEVEL ONE (END USER) NETWORK
CAPACITY PLANNING DOCUMENT

***

LEVEL TWO (BOUNDARY LINK) NETWORK
CAPACITY PLANNING DOCUMENT

***

NETVIEW AND NETVIEW/PC TUTORIAL

Dr. Kornel Terplan - Communication Networks Consultant

Performance Navigation, Inc. - 28 Summit Avenue
Hackensack, New Jersey 07601

ABSTRACT

The integration of logical and physical network management becomes one of the challanges for managing communication networks. There are several instruments providing support to main- and sidestream network management, including network, modem, application and software monitors, data bases, and analytical, emulation and simulation models. The major limitation of all of the above is that none of them provide an integrated solution. The tutorial offers an in-depth technical review of NetView and NetView/PC for addressing integrated network management. This include such areas, as support of the SNA products NCCF, NLDM, NPDA, LPDA, TARA, NMPF, ALERT, TAF, VNCA within NetView. It also includes the integration of LENs, LANs, PBXs and third party architectures utilizing alert, operator, host data and services point command facilities of NetView/PC. In addition, application program structures, panel hierarchy, code and command customization, files architecture and management, selection of information exchange alternatives, concert of real-time and PC-DOS components of NetView/PC are also addressed. To maintain objectivity, the paper also discusses the limitations of NetView and NetView/PC due to APPC, NMVT, DDM and CICS constraints. Other issues discussed are the cooperation of third party vendors, experiences yielded by the initial implementations as well as future trends.

1. INTRODUCTION

The need for adequate service levels in complex communications networks drives the demand for fast, consistent, expert-quality response to network management problems. Effective, flexible automation of both routine and creative activities would satisfy this demand by improving network management productivity.

Network management means deploying and coordinating all the resources necessary to plan, operate, administer, analyse, evaluate, design, and expand communication networks to meet service level objectives at all times at reasonable costs with optimal capacity.

In order to improve network management quality, there is a definite need for the integration of different information sources, products and procedures. The present status of network management can be characterized as follows:

- Fragmentation by products (communication software, modems, matrix switches, local area networks, multiplexers, private branch exchanges)
- Separate logical and physical network management
- Lack of correlation of different information sources
- Separation by network architectures
- Gateways do not yet support network management
- No adequate control of end user service level
- Too much information collected
- Redundancy in products
- Lack of powerful configuration data bases.

NetView and NetView/PC may mean the first important step towards centralized, standardized and integrated network management for multiple communication forms.

Although IBM has offered network management applications software packages for many years, NetView represents the first time that these separate SNA host network management packages have been put together in a single product. NetView consists of the NCCF, NLDM, and the NPDA network management products, and also includes functions from the VNCA and NMPF products. IBM products currently supported by NetView include the 3720 Communications Controller, the 3725 NCP Token-Ring Interconnection, the 3728 electronic matrix switch, and the 586X modems.

IBM's other recent announcement is NetView/PC, a multitasking personal computer subsystem that can be used in conjunction with NetView to provide services that extend Communication Network Management (CNM) support to non-IBM communications devices. CNM support will be extended to IBM Token-Ring Networks, non-SNA communications devices, and voice networks (CBX/PBX). The basic services provided by NetView/PC include services available to IBM or Rolm device-dependent applications, services to a user-written IBM PC-DOS program, and services to a technician/operator.

2. NETVIEW

NetView uses IBM`s SNA Network Management Architecture that supports the capabilities of new device attachments and permits flexible network growth. Architectural capabilities include Alerts, Session data, and Response time measurements. NetView supports any terminal in the IBM 3270 family with a minimum 24-line by

80-character display that is used as a NetView operator station when logged on through VTAM. The databases of the NetView components can use any IBM Direct Access Storage Device supported by VSAM.

2.1 NETVIEW ARCHITECTURE (Figure 2.1) (DATA87)

The functions of NetView are seen in the components that make up NetView, which include the following: (TERP87)

- Command facility - includes and enhances the functions of the Network Communication Control Facility (NCCF)
- Session monitor - contains the functions of the Network Logical Data Manager (NLDM)
- Hardware monitor - includes the functions of the Network Problem Determination Application (NPDA)
- Status monitor - allows network operators to look at the status of network resources and to choose commands to be sent for any displayed resource
- Online Help facility - supplies data about NetView commands and many of the NetView Display panels
- Help Desk facility - offers an online guide about network problem diagnosis techniques
- Browse facility - allows operators to look through libraries and the network log

2.2 NETVIEW SUBSYSTEMS

The most important subsystems of NetView are discussed in some depth (TERP87).

The virtual telecommunication access method (VTAM) has its own command language and operator functions that allow commands and messages to be issued and received via the system console. In the past, the sharing of the system console for VTAM and operating system messages created operational problems for IBM customers with large networks. To alleviate this problem and to separate network messages from host-system messages, an interface called the programmed operator interface (POI) was made available in VTAM.

The network communication-control facility (NCCF) is the current IBM product that uses POI. With this interface, terminals defined to NCCF can issue any command to VTAM that can be issued from the system console. The terminal can also receive messages from VTAM and output them to the NCCF terminal. In addition, NCCF uses the communication network-management interface (CNMI) provided by the SSCP.

NCCF allows grouping of individual commands into lists, called CLISTs, that can be invoked with one short command. NCCF provides program-based services for user or IBM-written applications. These services can be used for communication network-management applications or to tailor operator control functions.

In multiple system networks (MSNF), NCCF provides for communication across domain boundaries. Coordination and control of the network can be assigned to NCCF operators in any or all of the domains in the network. NCCF may become the central point for analyzing and filtering all SNA-related messages.

The VSE/OCCF (operator communication-control facility) extends the control capability of NCCF for the inclusion of DOS/VSE control, operated at remote locations. MVS/OCCF allows one or more remote MVS systems to be operated from a host MVS system. It resides in the remote systems and optionally in the host. ROCF resides in the remote facility.

Network operational control is frequently over loaded by supervising various applications (IMS, CICS), various systems (MVS, DPPX) and network-related information (NETVIEW, NCCF, NPDA, VNCA, NLDM). Until recently, for each area indicated, different display devices were required for continuous control. To address the access-to requirement, terminal-access facility (TAF) connects to a variety of display-oriented network-management tools that reside in a variety of software bases. It ensures that messages are not lost while the screen is dedicated to a specific application. Both solicited and unsolicited communication are supported, and a correlation of requests to replies is provided. But, TAF is restricted to the use of operator control sessions only. Multiple application single screen (MASS) allows an operator to have up to eight sessions to different VTAM applications at a time. Screen support may be provided by 3290, or Series/1 or 3270 devices.

NPDA assists network operators in performing network-problem determination by collecting and interpreting records of errors detected within a network and by recommending possible resolution. NPDA is a NCCF IBM-written command processor with access to the accumulated error-data and statistics-data base. This data is presented as follows:

1. Error counts associated with communication controllers, channels, modems, terminals, and links.
2. Specific data, an error description, and a probable cause associated with each class of errors collected. Recommended actions to resolve the displayed errors are provided.

Two new protocol commands have been implemented via the communication network-management interface: request maintenance statistics and record-formatted maintenance statistics.

Request maintenance statistics are forwarded by the system service control point to an SNA node upon request of NCCF. NCCF issues a request when directed by the NPDA command processor. NPDA sets up the request sequence whenever an NCCF operator requests, via network node name, to view the problem-determination data for a specific SNA physical unit. This request is issued from a terminal connected to NCCF.

NPDA provides error-description and probable-cause statements on displays to the NCCF operator. The error descriptions and probable-cause statements are derived from an analysis of network error data stored in the NPDA data base. The problem-cause definition identifies the most likely network component or group of components that caused a failure of hardwa-

re or software.

NPDA allows the NCCF operator to receive advanced warning when a communication-controller data link connecting a specific station exceeds an error-threshold rate on the link. The operator is alerted if the ratio between traffic and temporary station errors exceeds thresholds, specified by the user, on a line basis. The alerts are designed to flow unsolicitedly from the network elements using record formatted maintenance statistics (RECFMS) Type 00 response unit.

The alert originator communicates the alert to the alert processor, which forwards it to the alert manager. The originator and processor may be combined in the same network resource. The manager may or may not coreside with the alert processor. Besides physically oriented alerts (e.g., stations, lines, etc.), application alerts generated by customer-specified error-performance limits being exceeded are also included.

Via the event-statistical recording filter, irrelevant events grouped by unit name and type can be supressed. The alert-generating filter enables operators to classify certain types for alarming. These are stored in a separate part of the NPDA data base, accessible for operational control at any time. The NCCF operator-viewing filter enables network management to set up authority levels for operational control. Finally, the NPDA customer-viewing filter assists operational control to customize the filtering characteristics for the specific tasks under consideration. NPDA provides a data base for further processing by other tools, such as Information/Manager, SMF, Statistical Analysis System (SAS), or MICS. The data copied onto performance data bases are very valuable in supporting diagnosis in on-line or off-line modus.

Until recently, the stations attached to a controller were not explicitly known to the SSCP. For satisfying the downstream requirement for subnetworks, a threshold analysis and remote access (TARA) program was developed as an additional NPDA feature. Information may be solicited by the operator but more frequently is driven by automated procedures. This information can be centrally displayed in a hierarchical form, consistent with NPDA. Interfaces for management reporting via Service Level Reporter (SLR) and Info/System are supported as well. Recently MICS offered an NPDA interface as well.

In order to faciliate and simplify the network operational control activities, VNCA (VTAM node control application) offers additional features under NCCF to NPDA. It provides full-screen color displays to monitor the status and current activity of all resources in the domain under consideration. Customer-friendly interfaces have been designed for easy communication with the network. More importantly, certain routine-reactivation sequences are automated, allowing network operational control more time to handle unexpected, more serious network problems.

Network logical data manager (NLDM) was designed in response to logical problem-determination requirements. NLDM is an application under NCCF for collecting SNA session-related information, correlating data related to SNA sessions between network-addessable units, and displaying relevant information at an NCCF operator station. The intent is to establish session awareness and provide session-trace capability. By collecting relevant activities in progress at and just prior to failure, central problem determination can be significantly faciliated.

The principal NLDM features are

- NLDM is an on-line, interactive software monitor that extracts information from the telecommunication-access methods (e.g., VTAM or TCAM) and from the network-control program in single and multiple domains for supporting logical network management.
- Data are displayed in full-screen mode, utilizing color support.
- It provides useful data (SNA protocol, data flow, Path Information Units (PIUs) and selected NCP control information) when network problems (e.g.,"hung" sessions and stopped applications) occur that otherwise might not produce problem related information about physical networking resources.
- The operator can select the session(s) to be traced prior to and during the sessions.
- Active session data is maintained in virtual storage. When the session is terminated, the data are transferred to a VSAM data base for further processing and later reference for accounting, problem determination, availability reporting, and capacity planning.
- NLDM provides information for local or remote resources in the same domain or in cross-domain sessions.
- Reduction or elimination of the need to recreate software problems or to run dedicated traces because of the ability to collect and store relevant SNA information in progress and prior to a failure.
- Support for the response time monitor (RTM).

NLDM information may be further processed and reported by MICS.

2.3 NETVIEW INTERFACES

In order to improve end user productivity, NetView is expected to provide interfaces for:

- Info/System (IBM) for supporting problem, configuration and change management
- Service Level Reporter (IBM) for supporting service level evaluation and reporting
- MVS Integrated Control System (Morino) for contributing to standardizing the performance data base, reporting and for faciliating correlation among various information sources
- Teleprocessing Network Simulator (IBM) for supporting script generation for quality

testing new applications and stress testing communication networks

- SNAP-SHOT (IBM) for providing input data on traffic volumes for improving the accuracy of simulation runs

- Best/1-SNA (BGS Systems) for supporting network wide modeling.

At the moment, it is still early to say what systems and in which sequence are going to be supported.

2.4 NETVIEW BENEFITS TO USERS

Besides combining separate functions into one program, NetView also offers enhanced capabilities in the following areas:

- Terminal Access Facility support of large screen and color applications

- CLISTs driven by application messages

- Disk log enhancements

- Modem support of IBM 586X Models 2 and 3 and IBM 5812

- New and generic alerts

- New "Purge Attached" command

- Token-Ring network support

- Virtual route blockage indication

- Session setup failure notification

- Extended Recovery Facility in MVS/XA

- Automatic operations and recovery

- Realtime update of the domain status panel

- Important message indicator

NetView also provides improved support of network management functions through learning aids, consistent use of color, common program function keys and procedural conventions, and ease of movement between components. Learning aids consist of an Online Help Facility, a NetView Operation Primer and the NetView Operation Scenarios publications, and an IBM PC diskette tutorial.

To simplify installation of NetView, customized procedures and installation aids for all NetView components are available, such as the 4700 support facility on MVS. An installation "cookbook" and a post installation verification procedure are also provided.

2.5 PROBLEMS AND RISKS

Besides of substatial benefits toward integrated network management, there are still certain drawbacks with the NetView implementation:

- NetView does not eliminate or reduce the information capturing overhead of subsystems, such as NLDM and NPDA.

- Network Performance Monitor (NPM) is no part of NetView. Thus, relevant information on NCP-slow-down, Byte count, Line utilization statistics, and the TPNS-Interface are missing.

- End user response time measurement is by means of RTM and NLDM, only.

- Continuous threshold supervision via NPDA requests too much resources.

- The LPDA-related functions require the use of IBM modems.

- No message filtering.

- Lack of interfaces to other products, such as Info/System, SLR and modeling instruments.

- NetView is not yet user-friendly at all.

- Help-desk support leaves a number of unanswered questions:
  x Is expert system support expected ?
  x How easy is to interrogate the existing documentation ?

- Netview is not unexpensive; all software around Netview require the latest version.

- It is not yet clear, how independent products, such as Mazdamon, Netspy, Net/Master, application monitors and host monitors are able to communicate with Netview.

2.6 COMPETITION OF NETVIEW

The only product available at the host level for doing similar jobs is Net/Master (CINC84). It is a multicomponent system designed to assist in managing and optimizing the performance of complex SNA networks. Every node can be connected within the network, and information can be shared by hundreds of end users. Net/Master is a multitasking, multithreading VTAM application that is designed to support a large number of concurrent users with minimum system overhead. Net/Master has been designed to be easy to install and is extremely user-friendly, utilizing a large number of full-screen panels and menus to facilitate function selection as far as possible. The product solves the problems of managing multiple applications on the same mainframe, supporting a mainstream approach. NCCF is not a prerequisite for running Net/Master.

Net/Command (AVAN87) uses a sidestream approach for integrating different information sources, such as application, software, communication, network, modem monitors and LAN/PBX servers, offering virtual operating and filtering features. Similar integration options are supported by other companies such as AT&T, Codex, Infotron and Timeplex.

3. NETVIEW/PC

NetView/PC is a multitasking personal computer subsystem that supplies the facilities to support communication of network management data between a personal computer and NetView on a host. NetView/PC was designed to be used in conjunction with NetView to provide servi-

ces that permit user-written programs to extend Communication Network Management (CNM) to non-IBM communications devices. Using NetView/PC, CNM support can be extended to non-SNA communications devices, voice networks (CBX/PBX), and IBM Token-Ring Networks. It offers the basic services needed by a device dependent CNM application program and a network operator.

3.1 NETVIEW/PC ARCHITECTURE (IBM87)

NetView/PC is a new product designed to extend IBM's network management into new customer environments and to help provide, in more and more networks, total enterprise-wide network management. NetView/PC can perform important functions and, at the same time, communicate information to NetView so that it can execute additional functions. NetView/PC supports stand-alone network management as well as complementing NetView to extend the scope of centralized network management.

NetView/PC applications support IBM Token-Ring Networks and voice networks. In addition, the Application Programming Interface/Communication Services (API/CS) functions provided by NetView/PC will enable customer-developed applications to support a variety of other devices not supported by IBM's current network management.

Netview/PC provides a diverse set of functions. The capability to execute different tasks concurrently enables foreground and backgound processing to share a processor's resources, increasing its total capacity. A co-processor communication card and supporting software provide both synchronous (SDLC) communication with the IBM host and asynchronous communication with ROLM voice network devices, while off-loading a significant amount of work from the PC processor.

Local problem determination provides management of device and software alerts (errors) at the Netview/PC console. Host alert transfer to NetView extends centralized network management to devices not currently supported within the SNA network. File transfer between NetView/PC and the IBM host, using LU 6.2 protocols, enables a wide range of information to be exchanged with the IBM host CICS/DDM (Customer Interface Control System/Distributed Data Management) product and thus made available to host application programs for reporting, billing, or other processing.

Local operator control of NetView/PC allows operator access to various functions. Using remote console support, NetView/PC can be controlled from another NetView/PC, or a remote operator can review the keying of the local operator. The DOS partition support enables most DOS programs to execute concurrently with some NetView/PC tasks. Finally, a command facility provides a command-response capability between the NetView operator and applications running on NetView/PC.

Three NetView voice-network accounting programs reside and execute in the host (Figure 3.1).

One of the critical challenges facing the NetView/PC product was how to provide all of the NetView/PC functions, including host communication, and still meet certain storage and performance requirements. To provide host communication, it was necessary to support LU (Logical Unit) 6.2 sessions for sending data files, and PU (Physical Unit) 2.0 sessions for alert processing. This requirement was met by using the functions in the Advanced Program-to-Program Communications for Personal Computer (APPC/PC) program in the co-processor card.

APPC and the remaining NetView/PC functions could not be loaded in DOS memory, however, because of storage requirements. The solution chosen was to off-load all the communication software, such as LU 6.2, PU 2.0, SDLC (Synchronous Data Link Control), and ASYNC (asynchronous protocol) to a co-processor. For this, A Realtime Interface Co-Processor (ARTIC) card was selected (Figure 3.2). This provided NetView/PC with the storage and operating cycles needed to execute all of the input and output processing without drastically affecting PC performance.

The Realtime Control Program (RCP) is an IBM supported program that executes on the co-processor. The multitasking operating system can control up to 253 concurrent tasks, each task executing on any of 255 possible priorities. The Realtime Control Program also provides various other functions that simplify the writing of tasks such as:

- First-level interrupt handlers. The program receives all software and hardware interrupts, which are then passed to the task interrupt handler.
- Resource handler. The program manages various resources provided by the co-processor, such as hardware and software timers, queues, memory, communication ports, DMA (Direct Memory Access) channels, and interrupt vectors. A task can request temporary control of these resources.
- Interface functions. These routines allow any application task running on the co-processor to interrupt the PC in order to exchange data and control bytes.

In addition, there are various program functions provided that run on the PC for support of the co-processor, including system initialization (power-on, self-test, interrupt vectors, and hardware timer initialization), loading, PC tasks to the co-processor interface, and asynchronous error handling.

3.2 NETVIEW/PC SUBSYSTEMS AND COMMUNICATION WITH NETVIEW

The support of open application programming interfaces allows customers to use the services provided by NetView and NetView/PC to provide consistent management of network devices, enhancing their ability to manage a multi-vendor network. Figure 3.3 illustrates the open application programming interfaces that NetView and NetView/PC support.

NetView/PC provides a set of open application programming interfaces known collectively as the Application Programming Interface/Communication Services (API/CS). Four

functions are supported to assist customer and vendor-supplied applications in providing SNA network management support for non-SNA devices, such as terminals, modems, PBXs, multiplexors, LANs, other architectures and matrix switches. The four separate functions in API/CS are Host Alert Facility, Operator Communication Facility, Host Data Facility, and Service Point Command Facility.

The Host Alert Facility (HAF) allows an application running in NetView/PC to build an generic "alert", based on information received from the machine the application is monitoring, and route it to the NetView/PC. The Alert Manager passes the alert to the local NetView/PC operator and sends it to NetView in the host, where it may be displayed or used to initiate a recovery.

The application uses the Operator Communication Facility to signal the local NetView/PC operator. This is particularly important when more than one application is executing in NetView/PC and one of the background applications needs to communicate with the operator.

The Host Data Facility permits NetView/PC applications to transfer DOS files to a CICS/DDM (Customer Information Control System/Distributed Data Management) facility. A CICS/DDM application may then alter or transfer the contents of the DOS files.

Conversely, the CICS/DDM application may send files to the NetView/PC application.

Three types of communications using an SNA host are supported by NetView/PC. The first type is the forwarding of alerts from NetView/PC to NetView in a focal-point host. The application creating the alert provides the routing instructions; NetView/PC then routes the alert to the NetView/PC operator or to Net View. This alert structure follows the NMVT (Network Management Vector Transport, a subset of SNA) format and flows over an SSCP-PU 2.0 session path to NetView.

The second type of host communications is the transfer of data files between the host and NetView/PC, using a Logical Unit 6.2 session. This session is supported at the host end by a Customer Information Control System/Distributed Data Management (CICS/DDM) transaction program. Files can be transferred in either direction, which can be initiated by the NetView/PC operator or by any application in NetView/PC.

Finally, the Service Point Command Facility allows a command issued from NetView to be routed to an application running in NetView/PC, where the command will be executed. The functions supported by the application via this facility are determined by the application developer and the capabilities of the device that the application manages.

3.3 NETVIEW CUSTOMIZATION

Most of the open application programming interfaces in NetView were present in the earlier IBM network management products. However, these interfaces, in combination with the NetView/PC API/CS facilities, permit extension and customization of network management for all devices in the user's network.

NetView supports open application programming interfaces via four major sets of facilities: command lists (CLISTs), user exits, command processors, and alert screen customization.

The Command Processor Facility allows a customer to supply additional commands that may be invoked by NetView operator or CLISTs. Command processors can communicate with non-IBM and non-SNA network devices via the NetView/PC Service Point Command Facility. It is through this facility that customers and vendors can extend various command and response facilities to non-IBM and non-SNA devices.

The CLIST Facility allows a customer to define a set of NetView and access method commands to be executed based on some triggering event, such as an operator command, message, alert, or expiration of a timer. The CLIST can issue a command that invokes a command processor to communicate with a NetView/PC application. In this way, a device`s operations can be automated.

User exists allow a customer-written routine to supress data that is not useful in this environment, put device-specific data into a form that may be worked on by NetView or a supplied application, and tailor the user interface by changing the commands and displays available to the operator. Exits also have the capability to initiate CLISTs or command processors in response to events reported to NetView.

An important function provided in NetView is the Screen Customization Facility, which allows a customer to define alert screens for non-IBM devices. In addition, a customer can change or delete specific alert screens for devices that are supported by NetView.

Through these four facilities, NetView can be used to centralize the control of vendor devices, integrating the management of both SNA and non-SNA devices. The command processor facility communicates with the customer`s devices. The exit and CLIST facilities allow the operation of these devices to be automated, while the alert-screen customization facility displays events about devices.

The application programming interfaces in NetView and NetView/PC complement one another. An alert issued by a NetView/PC application and sent to the host using the Host Alert Facility may be displayed using the alert-screen customization facility. Alternately, the alert could trigger a customer-defined action using the user exit and command processor facilities of NetView.

The emphasis of open application programming interfaces both in NetView/PC and NetView allow NetView's view of the network to be expanded. As customers participate in IBM's Open Network Management structure, NetView, NetView/PC, and customer applications will provide better enterprise-wide management of customer networks.

3.4 INFORMATION EXCHANGE BETWEEN NETVIEW AND NETVIEW/PC

This chapter demonstrates how alerts are escalated by Netview/PC to Netview and how Netview reacts. File transfer capabilities are addressed as well. The numbers of the steps in Figure 3.4 relate to the numbers in the following list.

An error is detected.

1. A User supplied network management application, executing in the DOS partition of NetView/PC, recognizes an Alert condition and calls DCJVAOO to send an alert to the hardware monitor.

2. NetView/PC logs the Alert and sends the Alert NMVT to the host.

3. NetView passes the Alert NMVT to the Hardware Monitor executing in NetView.

Reacting to the Alert.

4. The operator enters the user defined command with the target (NetView/PC application) name.

5. The command is recognized by NetView and the user supplied Command Processor (CP) is given control and passed the parsed input.

6. The CP checks that valid data is passed and then calls the user supplied subtask with the DSIMQS macro.

7. NetView passes control to the OST with the passed data.

8. The subtask checks the input and then builds an NMVT. The subtask then issues the DSIZCSMS macro to send the NMVT to NetView/PC.

9. NetView sends the NMVT to NetView/PC.

10. NetView/PC passes the message to the API/CS.

11. The user application calls DCJVCOO (API/CS call to receive a command) and the API/CS passes the message to the user application.

12. The user supplied application uses DOS to perform required communications with the device.

13. The user application processes the command and prepares a reply message and calls DCJVCOO (API/CS call to send a reply).

14. The user application calls DCJVO00 to notify the operator that the DOS Partition requires operator communications.

15. The API/CS turns on the DP icon on the NetView/PC operator display.

16. The API/CS sends the reply message to NetView/PC.

17. NetView/PC sends the reply to NMVT to NetView.

18. NetView passes the reply NMVT to the user supplied subtask. The subtask processes the reply and prepares a message to send to the NetView operator.

19. The subtask issues the DSIMQS macro to send messages to the user supplied CP to display to the NetView operator.

    Note: The subtask may display messages directly by issuing the DSIMBS macro.

20. The user supplied CP issues the DSIPSS macro to display the message data.

Transferring file data.

21. The user application calls DCJVO00 to transfer file data.

22. The API/CS passes the request to the Host Data Transfer program in NetView/PC.

23. The HDF initiates the transfer with the host CICS DDM application. The file data is sent to or received from the host.

3.5 APPLICATION DESIGN OVERVIEW

Using the API/CS

This segment describes the general applicability criteria of using API/CS. All further examples, however, will show the specific criteria for the Host Alert Facility, only.

An Application Request Block (ARB) is required for all calls to Application Program Interface/Communications Services functions. Storage for the ARB must be provided by the application program. An ARB should be dedicated to an API/CS function from the 'OPEN' to the 'CLOSE' of that function.

External Declarations (ie. EXTERN DCJVA00: FAR) for the API/CS library calls must not be in the application's code segment. All calls to API/CS subroutines are FAR calls.

To use an API/CS function, the application should be coded to:

1. Provide storage for an ARB for each API/CS function that the API/CS will use. The storage for the ARB should be dedicated to the ARB from the 'OPEN' of the API/CS function to the 'CLOSE' of the API/CS function.

   Each Application Request Block (ARB) is identified by an ARBID, ARBn, where n is a numeric character that identifies the function for which the ARB will be used. It is used by the API/CS to verify the start of the ARB and serves as an 'eye catcher' in a storage dump.

2. Check the address of the ARB in the AX and DX register pair when the API/CS returns control.

   The API/CS checks the ARBID and if the ARBID (ARBn) in the ARB is incorrect for the subroutine called, the ARB address is assumed to be invalid. The API/CS makes the AX and DX pair zero and returns immediately to the calling application. The

application must check the AX and DX pair for non-zero before they are used.

3. Open each API/CS function that the application will use.

4. Call each API/CS function as required.

5. Close each of the API/CS functions the application has opened.

The functions should be opened as part of the application's initialization process and all opened functions should be closed by the application's termination process. The application may call an opened interface as many times as is required by the application until the application closes the (API/CS) function.

The ARB contains a two byte request code field (in hexadecimal) that the application sets to indicate the function desired. The two byte Request Codes and descriptions are:

Request Code Description

Alerts

0101H Open the Alert API/CS

0102H Send an Alert

0104H Close the Alert

Host Alert Facility Example

This example shows design requirements and still open questions about processing alert messages third party devices by NetView/PC. The steps include:

- Generating unsolicited alerts by the third-party device; the format is completely different from each device, and from the required NMVT-format

- Transmission to NetView/PC

  The following items should be clarified:

  x Address of NetViews/PC port
  x Transmission medium
  x Transmission protocol (synchronous, asynchronous)
  x Transmission speed

- Receive by NetViews/PC

  x Allocation of receive parts and buffers
  x Syntax check
  x Allocation of internal message id.
  x Storing third-party alert

- Processing Alert in NetView/PC by the application program

  x Branching to the appropiate subsystem - (in this case: Host Alert Facility)
  x Format conversion onto NMVT
  x Storing NMVT and identifying the pointer to the NMVT

- Transmission to NetView by using SSCP-PU-services

  x Open alert API/CS
  x Send alert
  x Close alert API/CS

For successful management, building the NMVT plays the principle role. NMVT-s are referenced in the ARB-s (Application Request Blocks) A four byte field (Position 15, name: MVADDR) defines the printer to the Alert Major Vector which may represent multiple NMVT-s.

NMVT (Network Management Vector Table) has the following structure for Alarm Management:

- X`01` Date and time
- X`03` Hierarchy name list
- X`10` Product classification
- X`91` Basic alert
- X`00` Text message
- X`A0` Detail qualifier
- X`9F` Netview/PC alert subvector

One example shows the following structure:

Length X`0000`	X`01`	X`03`	X`10`	X`91`	X`10`
X`00`	X`9F`	X`A0`			

X`91' gives information on alert type such as permanent or temporary, on cause (e.g. hardware, software, microcode), on components impacted (e.g. control unit, modem, link, line) PBX, device.

For implementing NetView/PC, X`9F` subvectors have to be constructed very carefully. X`9F` contains all information about third-party-components.

The target records include:

- Alert description records
- Probable cause records
- User cause records
- Installation cause records
- Failure cause records and
- Recommended action records

This classification follows the existing NPDA-formats. These records are the targets for third-party-alarm-message-conversion.

As an example, the Alert Description Records for X`9F` subvector will be shown. The first step in application design investigates how third-party alarm records match with those shown in Table 1. As an example, Infotron 992 multiplexers are considered with the list of alarms (Table 2).

3.6 NETVIEW/PC BENEFITS TO USERS

Users, centralizing and integrating network management can benefit in many ways:

- Services available to IBM or Rolm device-dependent applications:

  x File manager, multitasking facilities, communications manager

  x Panel, keyboard facilities, display

- Services available to a user-written IBM Personal Computer DOS program:

  x Host file transfer and communication with NetView

- Services available to a technician/operator:

  x Operator control, remote console, general help

  x Problem management, service reminder, alert management

- Consolidation of alerting, billing and traffic information within one product running in an application host or a dedicated computer such as CMC

- Standardization of all alerting messages of various measurement and management instruments

- Correlation of information from various sources
- Opportunity of supporting one single interface towards modeling, reporting and data base systems.

3.7 PROBLEMS AND RISKS

Besides benefits of centralization and integration, there are still many unanswered questions. The most important open problems include:

- IBM may become - without any serious competition - the network management market leader controlling practically all third-party-devices

- Dependency of users from too many parties, including IBM, third-party-vendors (quality of network management) and who has implemented Netview/PC

- Request for implementing new software and customization of existing software

- Lead time up to the integration of all third-party-devices may take too long

- NetView/PC is no integrator, but just an intelligent protocol converter

- Third party vendors may decline of being controlled by IBM

- NetView/PC means redundancy, because the most third party vendors have been utilizing some degree of network management that may even be much richer than the NetView/PC functionality (e.g.: INMS from AT&T and UNMS from Timeplex)

- NetView/PC is not yet bidirectional; the question whether IBM permits third party vendors to gain access to control and/or to information

- Netview/PC may be overloaded when mass data processing is required.

- NetView/PC has serious storage and communication throughput constraints

3.8 IMPLEMENTATION EXAMPLES

The first two implementation examples are discussed in some detail.

- Voice Network Management

IBM and ROLM announced two communication network management software products that extend IBM's open network management architecture to voice networks. These new programs, which run in NetView/PC and in the host System/370 (Figure 3.5), allow central site operations to produce billing reports for telephone calls, choose cost-effective configurations for voice traffic, and trouble-shoot ROLM Computerized Branch Exchanges (CBXs).

The ROLM Alert Monitor offers network-wide alert monitoring and problem management at the IBM NetView host. A single Alert Monitor can serve a combination of up to 10 CBXs and CBX II 9000 nodes. The Monitor detects problem conditions, collects error and alarm infomation from the CBXs, and translates the information into one of the four types of alerts, categorized by degree of severity. These alerts are then passed to the NetView/PC alert manager for local problem management and can be passed on to NetView for centralized problem management. The Alert Monitor also can use information obtained from the ROLM Automatic Facilities Test System (AFACTS) to report changing trunk conditions.

The ROLM Call Detail Collector collects information from up to 10 CBXs or CC3 pollable storage devices connected to remote CBXs or selected non-ROLM PBXs.

The Call Detail Collector provides site collection, multisite polling, and host transfer of data, which may be used for billing, capacity planning, and network analysis. Information can be stored and later forwarded to an IBM host for processing, using NetView/PC's File Transfer Facilities (via Logical Unit 6.2).

The NetView/PC can be locally attached to the ROLM CBX via an asynchronous connection for collecting call detail records, and error and alarm data. This information can also be collected and stored by a locally attached storage device and later forwarded to NetView/PC via a dial connection initiated either by NetView/PC or the locally attached storage device. The connections initiated by NetView/PC can be scheduled to occur at regular intervals while connections initiated by the storage device will occur when storage capacity thresholds are exceeded.

The NetView Network Billing System is a host System/370 application that provides customers with a clear accounting of CBX costs and usage by departments, long-distance carriers, and equipment. The input for this application are the Call Detail Records generated by CBX, collected and forwarded to the host by NetView/PC. Tariff information is provided by the NetView Tariff Data Base for U.S. long-distance carriers.

The NetView Traffic Engeneering Line Optimization System is a set of five host application

programs that allow users to analyze Call Detail Records received from the ROLM Call Detail Collector program. These programs help determine cost-effective ways to configure a site's telecommunication system and also help identify problem areas within a customer's configuration.

- Token Ring Network Management

The IBM Token-Ring Manager Version 1.1 (Figure 3.6) is a program that assists the user in problem determination and error recovery on the Token-Ring Network. It can operate as a standalone application or using such NetView/PC applications as alert forwarding to host and remote-console support to provide total system network management. The Token-Ring Network manager monitors the token-ring network for hard and soft errors.

A soft error is recorded when an error threshold for transient, recoverable errors is exceeded. Soft errors may indicate a possible loss of a network resource. Hard errors may indicate faults in the token-ring network cabling system access units, or attached network adapters. Up to 260 devices on one ring are supported.

Network errors are recorded by the Token-Ring Network Manager in an event log on a disk or diskette. Other network events, such as stations joining and withdrawing from the network, can be recorded in the log to assist the user in network problem determination. Events logged may be displayed and printed for a specified station or period of time.

An alert condition is immediately signaled to the NetView/PC operator by an audible alarm and a highlighted indicator on the operator display. At the same time, an alert message can be transmitted over an SDLC link to NetView in the SNA host, using NetView/PC's alert manager and host communication services.

NetView/PC's remote console support allows an operator at a remote IBM PC to operate the token-ring network manager. The remote NetView/PC communicates with the PC attached to the token ring through an ASCII asynchronous link.

3.9 COMPETITION OF NETVIEW/PC

There are just few products on the market offering similar functionality. Examples are:

- INMS (Integrated Network Management System) from AT&T
- UNMS (Unified Network Management System) from Timeplex
- NMS (Network Management System) from Nixdorf
- INM (Integrated Network Management) from Infotron
- CONTROLNET from Unisys
- NM (Network Manager) from Codex
- Net/Command from Avant-Garde Computing.

The majority of the products provide value added characteristics in the particular domain and encourage third party vendors to accept the protocol for integration. Usually, the integrator offers an interface to NetView/PC or directly to NetView.

4. FUTURE EXPECTATIONS

With the introduction of NetView and NetView/PC, IBM is finally offering an integrated compatible line of software products. One of IBM's goals is connectivity among its products as well as non-IBM products. By opening up its communications architecture, it is allowing users to attach other vendors' products to the SNA network. A centralized source of control is another goal that IBM is striving for with these announcements. Previously, its network management packages were not necessarily compatible with each other and operators would have to jump from one application to another to gather information.

If NetView is the success that IBM hopes it will be, vendors of both hardware and software network management systems will face a major new player. Previously datacomm managers who had an IBM shop may have used another vendor's network management system to get statistical information on the network. IBM hopes to stop this by offering an integrated package that meets the users' needs. Vendors with independent network management systems may have to find new markets or go after the smaller shop.

One area that remains unclear is NetView's report-generating capabilities. Information on this ability remains sparse. Another potential problem is cost. NetView is expensive; it runs only on IBM mainframes, whereas other packages can run on less costly mini- or microcomputers. NetView/PC runs on the IBM Personal Computer, but to bring together all of the network management components, users will need a number of these personal computers. One other thought is that this is just the first step in integrating network management. Some of the products are not even available yet. What is currently available permits network managers to monitor the alarm conditions but diagnostic tests and reconfigurations cannot be done yet.

In summary, it is expected that the majority of third-party-network-management-vendors will seriously consider the implementation of the NetView/PC-gateway. In certain cases, users or external software companies will implement the gateways.

REFERENCES

(ATT87) AT & T.: Network Management System Specification, Morristown, New Jersey, 1987

(AVAN87) Avant Garde Computing, Inc.: Net/Command and Net/Adviser Specification, Mt.Laurel, New Jersey, 1987

(CINC86) Cincom, Inc.: Net/Master Users' Guide, Cincinatti, Ohio, 1986

(DATA87) DataPro Research Corp.: IBM NetView and NetView/PC, C38-491-101, Delran, New Jersey, 1987

(IBM87) IBM Corp.: NetView and Netview/PC Summary, Interface Vol.11, No.1, January 1987, Raleigh, North Carolina

(TERP87) Terplan,K.: Communication Networks Management, Prentice Hall, Englewood Cliffs, New Jersey, 1987

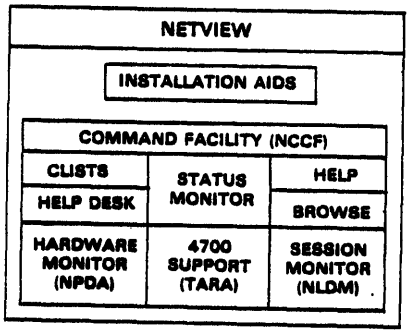


Figure 2.1; NetView Architecture

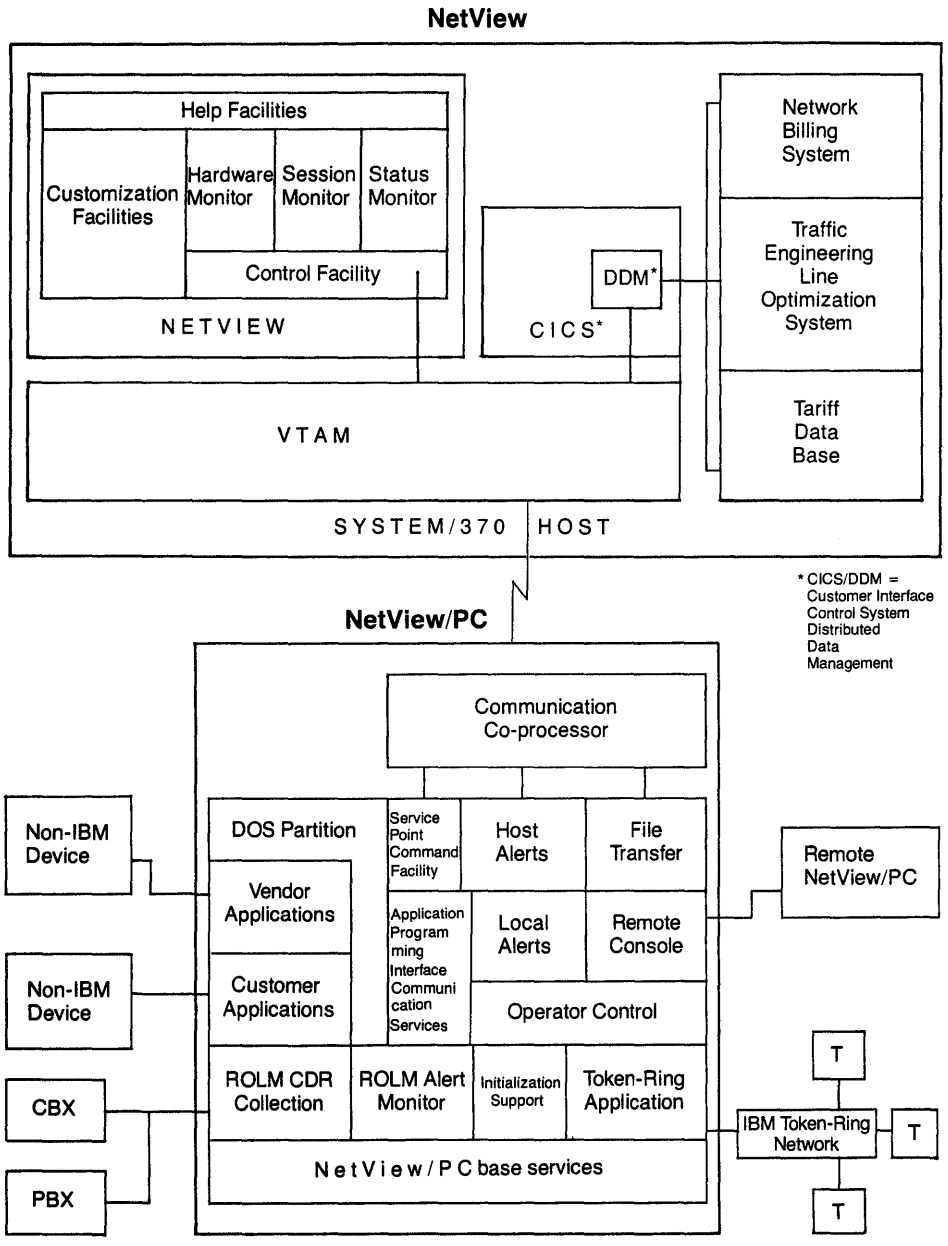


Figure 3.1: NetView and NetView/PC Structures

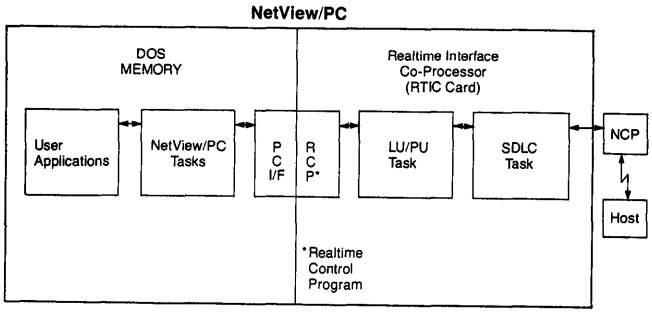


Figure 3.2: NetView/PC Realtime Interface

Co-Processor

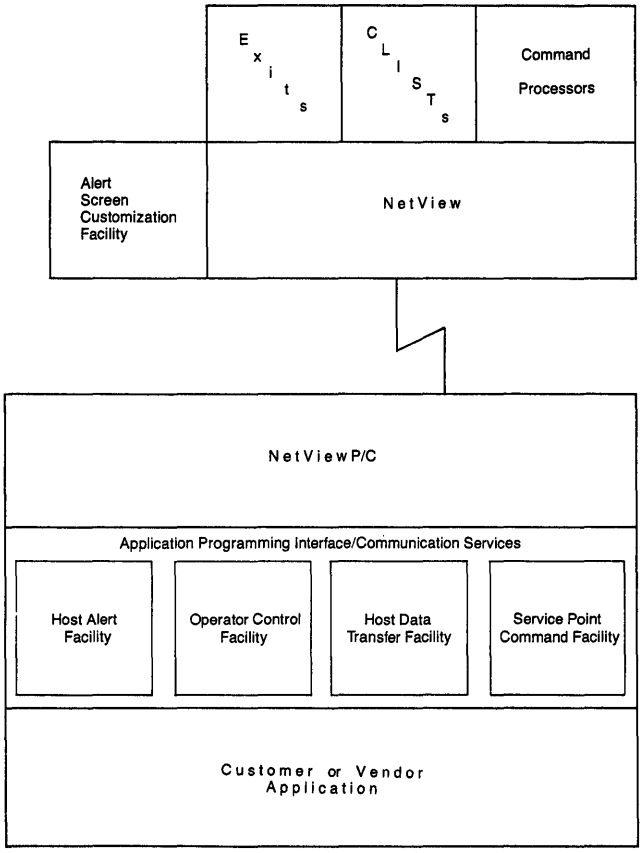


Figure 3.3: NetView and NetView/PC Open Interfaces

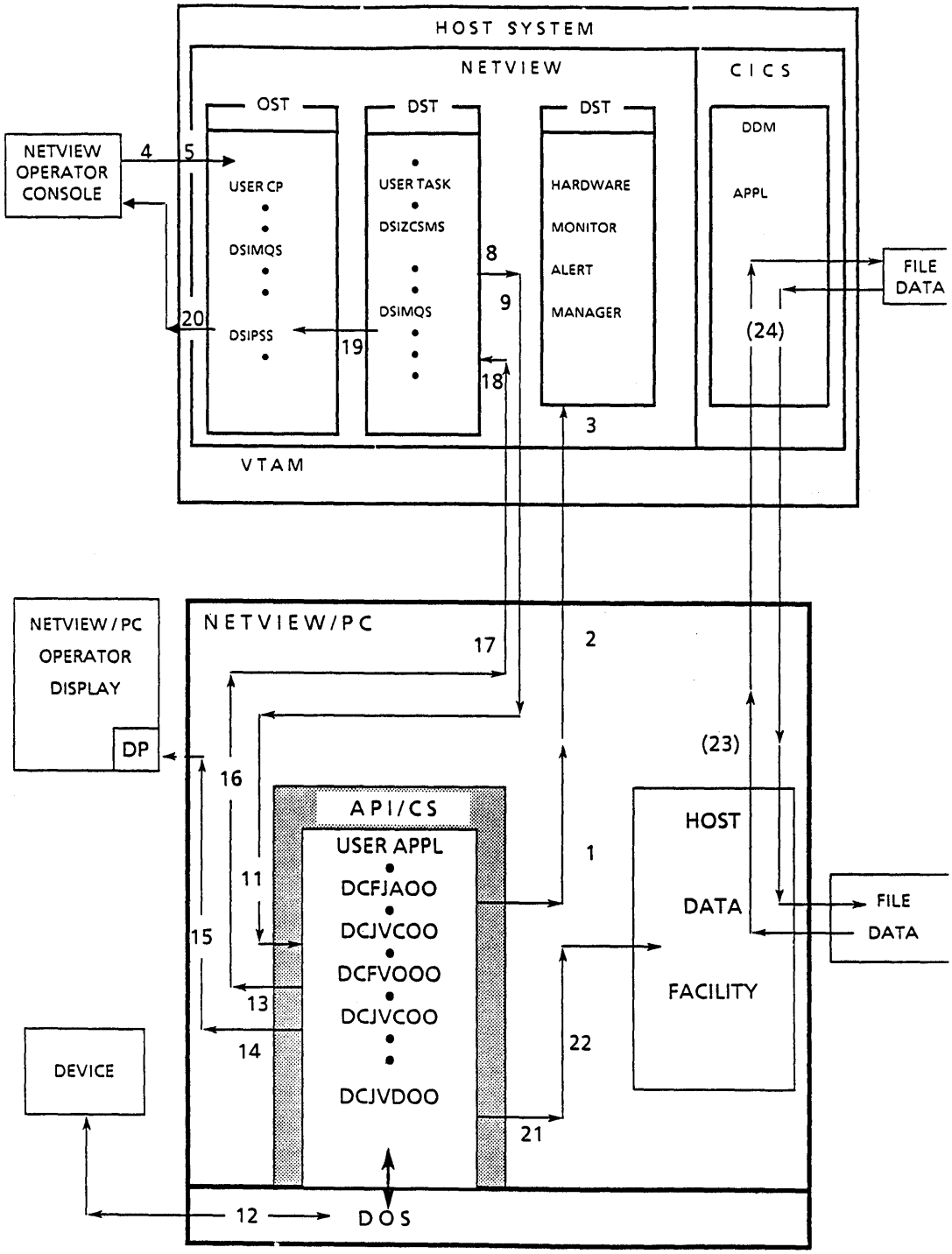


Figure 3.4: API/CS Flows Overview

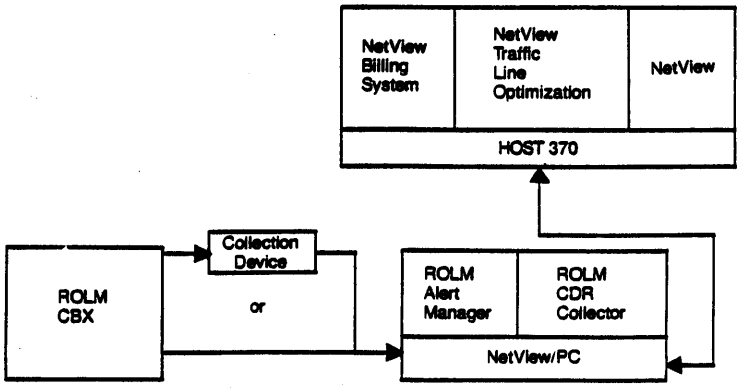


Figure 3.5: Application Support for Rolm CBX

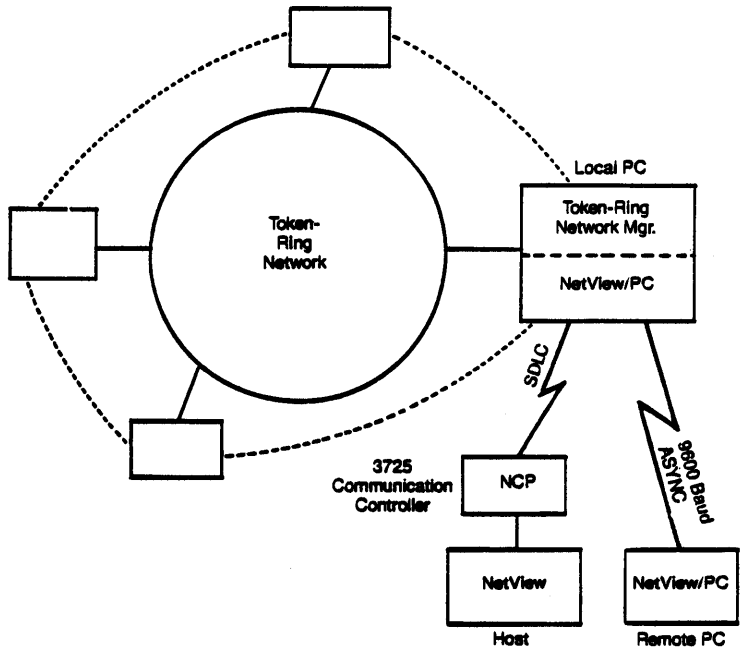


Figure 3.6: Token Ring Network Manager Support by NetView/PC

1. Buffer Full
2. Buffer 80% Full
3. Buffer was 80% Full; is now down to 60% Full
4. Channel Incapable of Flow Control
5. Clock Loss
6. Configuration Exceeds Burst Rate
7. Configuration Mismatch (with channel number)
8. Data Lost (flow control channel)
9. Data on Undefined Channel (with channel number)
10. Device Type Error
11. Exchange Bus Input FIFO Full
12. Flow Control Configuration Mismatch
13. Improper ABR Type
14. Improper Control Byte from Remote Link
15. Improper Configuration
16. Improper Post Byte
17. Inbound Line Error
18. Line Open
19. Line Restored
20. Local Carrier Loss
21. Local Initialization
22. Outbound Line Error
23. Primary Bus Read Fault
24. Primary Bus Write Fault

---

25. Primary Central Logic Reset
26. Primary Watchdog Timeout
27. Receive Queue Out of Sync
28. Remote Carrier Loss
29. Remote Initialization
30. Secondary Bus Read Fault
31. Secondary Bus Write Fault
32. Secondary Central Logic Reset
33. Secondary Watchdog Timeout
34. Software Switchover Completed
35. Switchover Request Error
36. Transmit Queue out of Sync

Table 1: Alert Description Records for

X'9F' Subvectors

TYPE	COI ENT	PSEQ#	COND	6 through 67 (62 character message)
1	2-3	4	5	
01	0100	01	FF	ABNORMAL TERMINATION
01	0200	01	FF	ACCESS ERROR
01	0300	01	FF	ACTIVATION ERROR
01	0400	01	FF	ADDRESS ERROR
01	0500	01	FF	BEACON ERROR
01	0600	01	FF	BUFFER ERROR
01	0700	01	FF	BUS ERROR
01	0800	01	FF	COMMAND REJECTED
01	0900	01	FF	CONNECTION ERROR
01	0A00	01	FF	DATA READ ERROR
01	0B00	01	FF	DATA WRITE ERROR
01	0C00	01	FF	EQUIPMENT MALFUNCTION
01	0D00	01	FF	INTERVENTION REQUIRED
01	0E00	01	FF	LOST DATA ERROR
01	0F00	01	FF	NOTIFICATION
01	1000	01	FF	OVERRUN ERROR
01	1100	01	FF	PERFORMANCE DEGRADED
01	1200	01	FF	POWER LOSS
01	1300	01	FF	PROCEDURAL ERROR
01	1400	01	FF	PROGRAM ABEND
01	1500	01	FF	PROGRAM ERROR
01	1600	01	FF	PROTOCOL ERROR
01	1700	01	FF	SPECIFICATION ERROR
01	1800	01	FF	THERMAL ERROR
01	1900	01	FF	THRESHOLD EXCEEDED
01	1A00	01	FF	TIMEOUT ERROR
01	1B00	01	FF	UNDERRUN ERROR
01	1C00	01	FF	UNDETERMINED ERROR

Table 2: Alerts Overview provided by

Infotron Multiplexors

A SURVEY OF PERFORMANCE APPLICATIONS OF THE TELEPROCESSING NETWORK SIMULATOR(TPNS)

Richard C. Burke
Duke Power Company

ABSTRACT

The Teleprocessing Network Simulator(TPNS) is recognized as a tool for preparing accurate pre-production response time estimates of major new application systems, and for simulation of complex network configurations. This tutorial will describe how the flexibility of the TPNS architecture and the scripting language make TPNS also suitable for a variety of other performance applications.

This tutorial begins with a brief introduction to TPNS, followed by a survey of some of its performance related applications. Descriptions of actual simulations and summaries of the results are also provided.[1]

1. INTRODUCTION

IBM's Teleprocessing Network Simulator(TPNS) is software that provides simulation for many different network/ terminal types and configurations. TPNS can be run in essentially two modes. The direct attached configuration allows for the complete simulation of the terminal response time(host response time and all network and VTAM response times). The VTAMAPPL configuration is used when only the stressing of the host application environment is required. Executing TPNS with the VTAMAPPL configuration requires only the installation of the TPNS software on a host computer. However, executing TPNS with the direct attached configuration also requires one or more standalone 3705/3725 network processors, and additional TPNS software on each 3705/3725. Defining a TPNS simulation using the direct attached configuration can be extremely complex and time consuming. However, defining a simulation configuration for a VTAMAPPL simulation is relatively simple. Our company runs TPNS exclusively with the VTAMAPPL configuration.[2]

When run with the VTAMAPPL configuration, TPNS uses multiple VTAM application programs(actual VTAM APPL statements), which establish one or more LU to LU sessions between secondary logical units(SLU - usually the terminals being simulated), and one or more primary logical units(PLU - usually the application(s) under test, such as a CICS Terminal Owning Region or a PROFS system). The VTAMAPPL configuration also permits simulation of the PLU allowing actual network terminals to logon the SLU. The scripting facility of TPNS generates the input transactions that are sent to the system under test. TPNS monitors the send and receive timestamps, allowing computation of an approximate host response time. Figure 1.1 is a simplified schematic diagram illustrating a view of the VTAMAPPL configuration for a simulation for 10 terminals and three applications under test. The "VTAPX" units designate actual VTAM APPL statement(s) defined to the TPNS VTAM system(VTAMA). A one to one correspondence between the SLUs and the VTAM APPL statements is only required if the PLU(s) do not support parallel sessions. The dotted line between the two processors illustrates that TPNS and the application(s) under test can run on the same processor. However, if the simulation is a stress test, TPNS should run on a different processor.

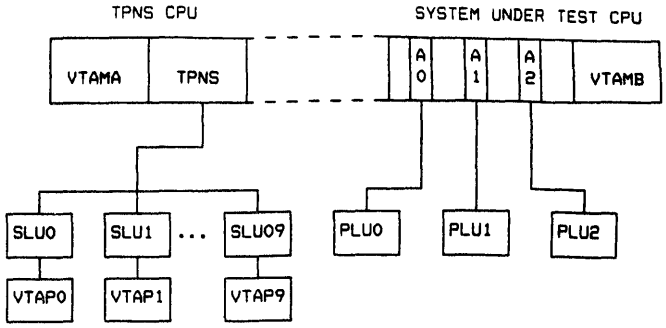


Figure 1.1

The specifications for a TPNS simulation are contained in the TPNS script. The script is made up of two primary components, the network definition(generally one per simulation), and the message text definitions (many per simulation). The network definition defines the environment for the test. This might include the attributes of the VTAM APPL ids, the attributes of the SLUs and PLUs(screen size, logical unit type, logmode entry, etc.), table data to be used in generating the input messages, the order in which the message text definitions will be executed, and overall parameters of the simulation(such as the default transaction think time). The message text definitions define the logic for generating the transaction input, testing the validity of the output, and controlling the flow of the test.

2. PERFORMANCE APPLICATIONS OF TPNS

The flexibility of the TPNS architecture makes TPNS suitable for a variety of performance applications. These applications include:

1. Stress Testing of Application or System Software
2. Performance Testing of Application Software
3. Performance Testing of System Software / Benchmark Creation
4. Functional/Regression Application Testing
5. Application Prototyping
6. Creation of an Expert System

---

[1] The views expressed in this paper are those of the author, based on his experiences using the TPNS product. They do not reflect the official policy of Duke Power Company.

[2] All performance related applications of TPNS discussed in this tutorial assume the VTAMAPPL configuration.

2.1 STRESS TESTING OF APPLICATION OR SYSTEM SOFTWARE

The objectives of a well designed application stress test are to determine the impact of a given application at peak arrival levels on the performance and capacity reserves of the system as a whole, and to determine if the application will adequately meet standards. There are two methods for meeting these objectives without the use of TPNS. The first method is to perform a stress test with the clients who will be using the system when implemented into production. Although this method does work on occasion, lack of adequate client training, lack of adequate test planning, and most importantly lack of test repeatability make this method prone to failure. The second method is to create an analytic model of the application under test with performance data from a similar production application, performance data from the test environment, or with an application modeling package, and model the impact of bringing the new or modified application into production. This method also can be very successful, at a relatively low cost, given the correct circumstances. However, for a totally new major application, a critical operational application, or a change in an environment with very low capacity reserves and margin for error, a well designed stress test performed with TPNS is often the best way to prevent major unexpected performance problems.

To ensure that the TPNS test actually simulates the production usage of the application, it is essential that the script be developed with the assistance of the application development and primary client areas. After thorough testing in a test environment, the stress test should be run in the production environment (production configuration, production DASD, etc.) in a stand alone mode. If the results from this test are successful(i.e., response of the application meets standards), an analytic model of the application should be built, and merged with a baseline model of the production system. If the resulting model demonstrates that the complete production environment will meet standards, a final stress test should be run with the application under test running with all other production workloads. The analytic model should then be calibrated with the arrival rates from the stress test. The model can then be confidently used for future capacity planning studies. This methodology applies equally to the stress testing of a new version of system software(operating system, subsystem, etc.), except the testing must take place in a standalone environment, and must simulate all significant workloads of the system/subsystem under test.

2.1.1 EXAMPLE. In this example, an application stress test was designed to determine if a newly developed corporate online payroll entry application could be successfully implemented into production within the service level specifications. The application was designed to run in the CICS 1.7 environment, using the DB2(primary DBMS) and ADABAS DBMS. The design of the application called for a peak transaction rate of 2 to 4 transactions per second from 250 to 400 terminal operators. The requirement that the entry of payroll data be completed within a short time window made tuning the response time of the utmost importance(the guideline was 96% of all transactions be completed with an end user response time of less than five seconds). The TPNS simulation called for the simulation of the entry of time sheets(one or more transactions for each employee), a summarization/balancing transaction, and a posting transaction. Performance issues discussed during the design of the script included concerns about possible delays due to the use of two DBMS systems, delays due to the possible overloading of an already loaded CICS Application Owning Region, and the possibility that the addition of this application would overload the cpu processor. Since this was the first major on-line DB2 production application to be installed, it was critical that a thorough simulation of the production environment be performed. These concerns identified the key points in the simulation design(simulation of a representative sampling of all data base files, simulation of the many entry formats, and a gradual increase in the transaction arrival rate over the duration of the test). Since the development of the TPNS script was performed by a performance analyst with no background in the application, close cooperation with the application team was essential. The primary impediment to the rapid testing of the script was conflicts between the TPNS script tester's need for a stable application environment and the application tester's need to make final changes to the application.

2.1.2 RESULTS. A dry run of the stress test was scheduled the night prior to the planned production test. Minutes after beginning the test, it was apparent that performance was unacceptable. Host response, that should be in the 1 to 2 second range, was averaging close to 10 seconds with as few as 10 transactions per minute. The problem was determined to be in the performance of some of the DB2 plans.[3] Changes were made to the DB2 plans, the application was rebound, and another dry run of the stress test showed that the changes were successful. The full production test was then begun, and gradually the transaction rate was increased to approximately 1.5 transactions per second. Suddenly, TPNS appeared to stop. Analysis of the TPNS logfile revealed that the application had returned an unexpected response, and TPNS had hung in a loop. Fortunately, the problem was quickly resolved, and the test was restarted.

Over the test period of almost an hour and thirty minutes, the number of simulated terminals was increased to nearly 200, at which point the one minute transaction rate was over three transactions per second. However, as illustrated in Figure 2.1.1, the transaction rate dropped significantly at several points during the test(indicating a serious performance bottleneck). Figure 2.1.2 shows a corresponding dramatic increase in response time at these same points. Although the response time was generally stable(except at the bottleneck points), it was clear that the application would not meet performance standards.

SAS code was developed to analyze the collected performance data from THE MONITOR FOR CICS(Landmark Systems Corp.). This analysis identified the following breakdown of the host response time:

TRANSACTION	DB2	ADABAS	RPL	MISC CICS
Primary Transaction	25-50%	40-50%	10-20%	10-15%
Average of Remaining Transactions	70%		25%	5%

From this data, three primary bottlenecks were apparent:

- o Delays due to DB2 processing
- o Delays due to ADABAS processing
- o Delays due to excessive CICS relocatable program loads(RPL)

---

[3] A DB2 plan is the control structure produced during the bind process and is used by DB2 to process the SQL statements required for execution of the application. The bind process includes compilation of the SQL statements, selection of the optimal access path, and output of the DB2 application plan.

Subsequent analysis revealed that considerable DB2 delays were caused by DB2 deadlocks resulting from locks being placed on an obsolete DB2 object; 50% of ADABAS delays were caused by duplicate opens and closes being performed on ADABAS accounting files; and the CICS RPL problem was resulting in considerable delays to the existing CICS workload. As a result of the information gained from the stress test the following performance recommendations were made:

- Remove the lock requests for the obsolete DB2 object
- Redesign the ADABAS routines to optimize the file I/O logic
- Set up a new CICS Application Owning Region for this application

Due to resource constraints, the ADABAS modifications were deferred. The other two changes were made and plans were readied for a second production stress test.

The results of this test were more encouraging. The peak transaction rate was consistently maintained between six and seven transactions per second (considerably higher than the estimated peak) with approximately 380 terminals. The host response of the most important transaction was consistently between 1.0 and 1.3 seconds. Some executions of the summarization transaction ran in excess of 30 seconds due to its selection of up to 5,000 DB2 rows. This was judged to be insignificant due to the low frequency of this transaction. During the execution of the test, the processor requirements of the application caused the processor to be consistently utilized at levels in excess of 100%, resulting in a significant backlog in the production batch and TSO workloads. A subsequent analytic model of the new production environment provided justification for the early upgrading to an IBM 3090/400 processor.

2.2 PERFORMANCE TESTING OF APPLICATION SOFTWARE

TPNS can be very useful in the identification of the component(s) of an application responsible for the highest level of resource consumption. This is especially valuable in an application such as PROFS where no response time data by transaction name is available. Scripting for this type of simulation is accomplished by coding several identical message text decks for the application. The decks are then altered to terminate at successively later stages in the deck. Subsequent analysis of the performance data will identify the component(s) in the application responsible for the performance problems. This technique was used in the "DW/370 vs. DCF" comparison described below.

2.2.1 EXAMPLE. In this example, the relative resource requirements of the two PROFS document text processors (DW/370 and DCF) are compared. The purpose of the simulation was to determine if the installation of DW/370 as an alternative document text processor would have a significant effect on the resource consumption of the PROFS workload. The values of several resource measures(transaction count, trivial to non-trivial transaction ratio, cpu time(virtual and cp), paging reads, non-paging I/O, working set size, response time, etc.) were collected to create a benchmark of the resource requirements of each text processor in a completely controlled environment. The script consisted of the PROFS input required to create or update, proofread, correct errors, review, print, and return to personal storage, one of three different documents(a 1½ page document, a 1½ page table, and a 5 page document) for both text processors. The script was designed using realistic think times and attempted to minimize the number of terminal interactions(use of the "ENTER" key). Each simulation benchmark (combination of document type, text processor, and text action such as create or update) was executed on a separate VM id to permit a clean analysis of the performance data. The computation of the resource benchmark was done by running the reduction and user classification phases of the IBM VM Performance Planning Facility(VMPPF) modeling software.

2.2.2 RESULTS The results of the simulation demonstrate that usage of DW/370 requires significantly more processor capacity, non-paging I/O, and considerably reduces the non-trivial to trivial transaction ratio(resulting in a corresponding increase in the number of non-trivial transactions). The increased resource levels are much more pronounced for the larger document(five pages) than for the smaller documents(1½ pages). Figures 2.2.1 and 2.2.2 illustrate the differences in processor and non-paging I/O(VIO) for the three simulated document types.[4] Although it was expected that DW/370 would consume more processor resources, the large increases in the non-paging I/O and non-trivial transaction counts were puzzling. To determine the cause of the unexplained resource increases, the TPNS script was modified by creating duplicate TPNS message text decks for the different benchmark simulations. Each deck was altered to terminate at a different point in the document conversation, to identify the resources required for each component. The simulation was rerun only for the long document, with unexpected results(Figures 2.2.3 and 2.2.4). The primary difference in resource consumption was in the document proofread and correction script(DW/370 consumed almost three times as much processor time and almost five times as much non-paging I/O as DCF). Finally the simulation was rerun with seeks analysis turned on. The results(Figure 2.2.5) again show that the proofread function was the primary performance bottleneck. The ratio of DW370 seeks to DCF seeks for the proofread/correction function was approximately 2.5:1 for the product mini-disks, but a whopping 27:1 for the user A disk.

2.3 PERFORMANCE TESTING OF SYSTEM SOFTWARE / BENCHMARK CREATION

TPNS can be successfully used as a vehicle for testing system performance tuning parameters. The major difficulty in this form of TPNS testing is the large number of workloads that must be simulated to create an accurate benchmark of typical system activity. One notable exception is a stand-alone VM system for office automation(PROFS). If a realistic benchmark can be created, the complete repeatability of TPNS can be used to study the impact of changing various system tuning parameters in a totally stand-alone environment.

2.3.1 EXAMPLE. In preparation for the installation of a new operating system for a production VM system (nearly 100% of usage is PROFS), a requirement was identified to perform a stress test on the new operating system, and create a performance benchmark, to assist in tuning the new operating system. The objective of the benchmark study was to design a script that when run would exhibit similar performance characteristics of the production system(in terms of processor utilization, paging/swapping, and non-paging I/O). Scripts were already available for several of the typical PROFS functions(document creation, calendar update, mail read, etc.). These scripts as well as several new ones were combined to form a PROFS benchmark script. The script included such features as realistic think times, wait scripts(to simulate periods

---

[4] For a more extensive analysis and listing of the results of the initial simulation, see reference [1].

of complete inactivity), and selection of the function script(open the mail, check calendar, etc.) based on the expected probability of that activity occurring in production. The completed script was run in a test environment with 50 PROFS ids in order to compare the processor utilization, non-paging I/O rate, and command rate per PROFS id with that which was occurring in the production PROFS system during peak activity. Based on the results of this comparison, new transaction think times were computed to allow for resource activity comparable to that expected for our production system 6 to 9 months from now. The final TPNS script was then run with 800 PROFS ids for a period of approximately 45 minutes under both the current and the new operating system. Based on the results of the test, initial tuning objectives were established. In the future, an expanded version of this benchmark script will be used to perform additional tuning, to predetermine the effects of the expanding PROFS community on the VM virtual storage constraint, to determine the break point(in terms of number of logged on users) of the GCS VTAM virtual machine, and to conduct other performance studies.

2.3.2 RESULTS. Available at presentation in December

2.4 FUNCTIONAL/REGRESSION APPLICATION TESTING

TPNS can be used in the application development functional testing process to ensure that all documented expected results are achieved. After all functional testing is complete, the functional test scripts can be modified and reused during the stress test phase, and whenever regression testing is required(operating system/subsystem changes or minor changes to the application). The major difficulty in this type of testing is providing the development staff with the required level of training to code the initial functional testing scripts. This difficulty can be overcome by the creation of a functional script generation program. The input for such a program would be the documented application expected results. The program would then output a runnable TPNS functional test script.

2.5 APPLICATION PROTOTYPING

It is possible to simulate the primary logical unit(the actual application) as well as the secondary logical unit(terminal input). Simulation of a PLU with TPNS provides the capability to allow a client to logon from any VTAM terminal and view the current version of the design of the application. Furthermore, the simulated primary logical unit can be used when developing the stress test script, so as to avoid conflicts between the development of the application and the development of the stress test script. This basic concept was used when developing a prototype of an "OMEGAMON Monitor Database" application that is described below.

2.5.1 EXAMPLE. The purpose of this example is to demonstrate the powerful capabilities of the TPNS prototyping concept, not necessarily to recommend the development of this particular example. This example demonstrates the methodology for the creation of an "OMEGAMON Monitor Database" application that was developed with the TPNS prototyping concept. The idea evolved during the development of several VM/PROFS performance simulations. The need was identified to document detail performance data from the OMEGAMON/VM (Candle Corp.) performance monitor over the duration of the simulations. This requirement was met by designing a script to logon a simulated terminal to OMEGAMON/VM, and logging the results to the TPNS log file. This script was then included in the actual VM/PROFS performance script to be run. Although this method permitted the retention of the desired OMEGAMON/VM displays for later analysis, the performance analysts viewing the simulation also needed to view the same displays. Although each analyst could logon to separate VM ids and invoke OMEGAMON/VM, this might result in a stress test of OMEGAMON/VM. It was decided that the prototyping feature of TPNS could be used to allow any analyst to logon to a TPNS simulated PLU which would display the last received OMEGAMON/VM status screen. This method would place no additional overhead on the system under test. The development with TPNS of a simulated PLU requires only a basic knowledge of TPNS scripting and a general knowledge of 3270 programming concepts(TPNS takes care of the SNA protocols of sending and receiving data between the simulated PLU and the actual terminal). An implemented version of this methodology was used during a benchmark test of a VM/PROFS system(described in more detail in section 2.3). An expansion of this methodology was used to build a prototype of a system-wide "OMEGAMON Monitor/Database".

2.6 CREATION OF AN EXPERT SYSTEM

The scripting features of TPNS that allow checking for application expected results make TPNS especially well suited for use in the rules testing phase of an expert system. The advantages of this concept are that an expert system for any complex software can be built by simply defining the rules base, providing a mechanism for collecting input(an ISPF panel, etc.), and coding the rules base within a TPNS script.

2.6.1 EXAMPLE. The purpose of this example is to demonstrate the broad concepts of producing an expert system from any existing software product that solicits input in a full screen mode. In this example, a prototype of an expert system for the IBM VM Performance Planning Facility(VMPPF) is described. VMPPF is software used for modeling performance and capacity planning configuration changes in the VM environment. VMPPF is quite adequate for occasional usage by an experienced VM analyst. However, for developing numerous models or for usage by an analyst with limited VM expertise, the use of an expert system would be very desirable. The primary objective for developing this prototype was to develop a methodology that would eliminate all repetitive procedures, encode the knowledge base within the system, and permit the development and production of the results for multiple models with a minimum amount of analyst interaction time.

A basic flow of the expert system is as follows:

Invoke the system under either TSO or CMS

The primary ISPF panel is invoked(Figure 2.6.1)

The ISPF panel allows for the selection of the desired model building/execution options for this execution of the expert system(panel options 1-4), which produces a TPNS table file to be used in simulating VMPPF.

The execution of VMPPF via TPNS is requested. This marks the end of the user interaction with the system until execution of the VMPPF software is complete.

The execution of the TPNS simulation is begun. TPNS will simulate the requests for tape mounting, input of all predetermined data for VMPPF, and use of the rules base for determination of all other input to VMPPF.

After all requested VMPPF models are computed and reports are generated, the primary ISPF panel is reinvoked.

The selection of display or printing of the previously generated modeling reports/graphics is made, and the requested information is provided.

The expert system is terminated.

Figure 2.6.2 illustrates the expert system flow:

```
---------------------- VMEXPERT PRIMARY OPTIONS MENU -----------------
OPTION ===>

SELECT ONE OF THE FOLLOWING ACTIVITIES:

   0  ISPF PARMS         -  SPECIFY TERMINAL AND USER PARAMETERS
   1  DEFINE CLASSES     -  CREATE/EDIT USER CLASS PATTERNS
   2  DEFINE CALL RATES  -  CREATE/EDIT USER CLASS CALL RATE PATTERNS
   3  DEFINE PARAMETERS  -  CREATE/EDIT MODELING PARAMETER PATTERNS
   4  DEFINE REQUEST     -  CREATE/EDIT MODELING REQUEST
   5  EXECUTE REQUEST    -  EXECUTE DEFINED MODELING REQUEST
   6  DEFINE OUTPUT      -  CREATE/EDIT DISPLAY/PRINT REQUEST
   7  PRINT REPORTS      -  PRINT SELECTED MODELING REPORTS
   8  DISPLAY GRAPHICS   -  DISPLAY SELECTED MODELING GRAPHICS

   P  PRINT              -  CREATE/EDIT PRINT JCL

   T  TUTORIAL           -  DISPLAY VMEXPERT TUTORIAL

   X  EXIT               -  EXIT ISPF USING LIST/LOG DEFAULTS

HELP TUTORIAL PANELS MAY BE DISPLAYED BY ENTERING HELP ON ANY PANEL

ENTER END TO EXIT VMEXPERT
```

Figure 2.6.1

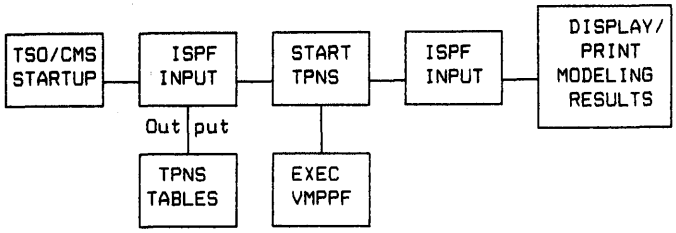


Figure 2.6.2

3. GENERAL TPNS SCRIPTING DESIGN RECOMMENDATIONS

Determine the objectives of the test before any work is begun on the script. The TPNS scripting language is extremely flexible. If one understands the major concepts of TPNS scripting, practically any realistic testing scenario can be simulated. TPNS also supports an extremely open EXIT architecture for options that the scripting language does not support.

Design the simulation on a "macro-level" instead of on a "micro-level". In other words, determine what level of system activity should take place. Do not be concerned with the specific transaction mix(and specific input) of one terminal. This allows all terminals to execute one set of message text decks. Judicious use of path distributions and data tables will enable a script to simulate the proper transaction and data mix respectively.

Design as much **flexibility into the script** as possible. Designing a script with only one set of message text decks permits the addition of any number of terminals without defining additional decks. This design substantially minimizes the complexity of the overall simulation. Another method for adding flexibility into the script is controlling the simulation with one "Master LU" deck. The script is made flexible by coding all possible control actions in "ON EVENTS", determining which LUs should be affected by a given control action with one or more network/device counters, and specifying which control action is to be performed next by signalling the required ON EVENT from the console.

Design as many **self-checking and recovery** features into the script as possible. Considering the substantial investment made in the development and testing of the scripting process, it would be disastrous if the test failed for lack of one of these features(a definite possibility).

Build a script library, so that scripts and techniques that have worked successfully in one test, can be reused in succeeding tests. This is especially important if a benchmark of a production system may be desired in the future.

Only use the **script generator facility** if there is a good reason for doing so. Using the script generator facility causes the test designer to in most cases lose control of the content of the script(one of the primary advantages of TPNS). Also, use of the script generator facility limits the addition of terminals to the test (added terminals will be executing exactly the same transactions with exactly the same input data as defined in the captured script).

Monitor the results of the test with the normal application monitors. Although TPNS captures data that permits computation of an approximate host response time, its value may differ considerably from that computed by the application monitors(especially for VM).

4. SUMMARY

The effective use of TPNS in the CPE area has the potential for giving definite answers to questions that without its use are left to educated speculation and qualified answers. This tutorial has identified six major performance applications with examples, for the use of TPNS with the VTAMAPPL configuration. Specific recommendations for the effective design of TPNS scripts have also been provided. It is the hope of the author that this tutorial will provide the incentive for more CPE professionals to make use of the less traditional applications of TPNS in the solution of complex CPE issues.

REFERENCES

[1] A. H. Shub, "PROFS Capacity Planning", Share 69, August 1987(Also to be in the "CMG '87 Conference Proceedings)

[2] IBM, Teleprocessing Network Simulator User's Guide, SH20-2490-2, June 1985

[3] IBM, Teleprocessing Network Simulator Planning and Installation, SH20-2488, June 1985

[4] Landmark Systems Corporation, THE MONITOR FOR CICS, Version 6.2, Users Guide, June 1986

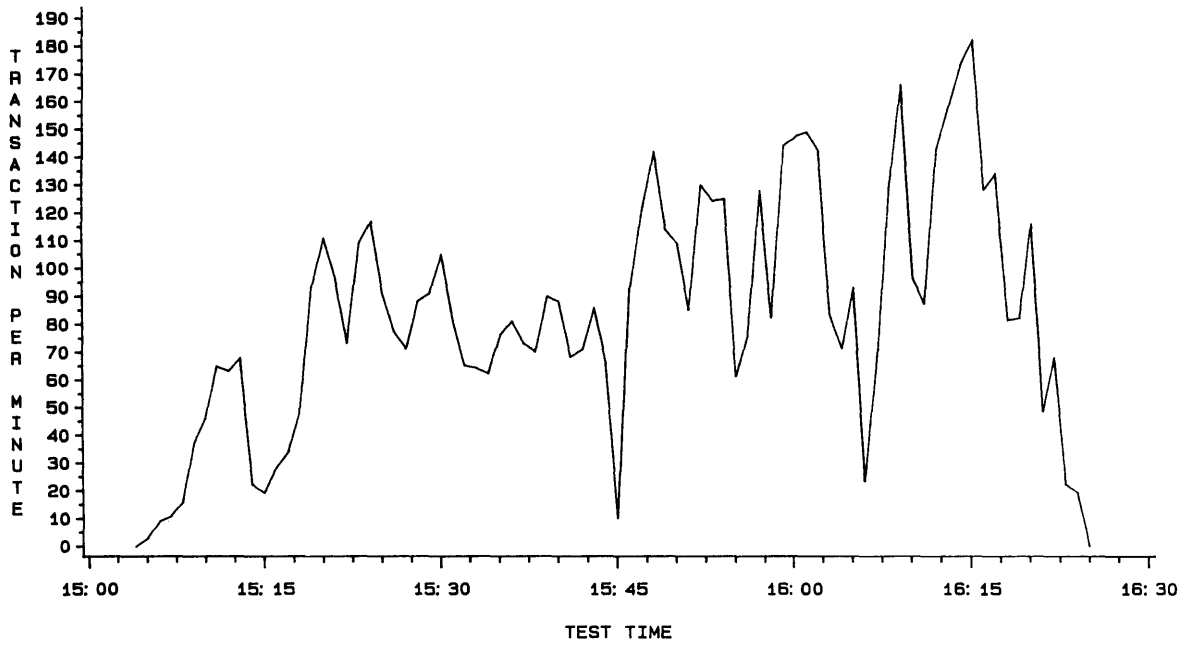
[5] IBM, IBM DATABASE 2 System Planning and Application Guide, March 1986

[6] DB VIEW, DB2 CAPACITY MANAGEMENT, 1985

[7] Candle Corporation, OMEGAMON/VM Version 300 Release Guide VM/SP, VM/SP HPO, and VM/XA, OV60-1504-20, December 1986

TRANSACTIONS PER MINUTE

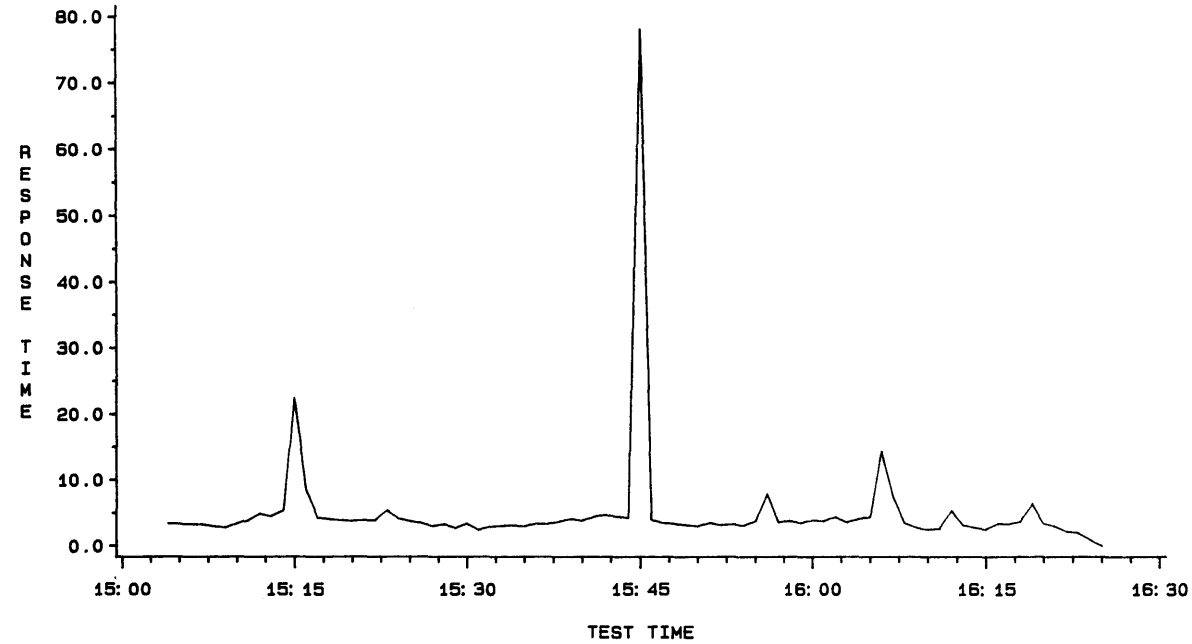
FOR FIRST CTR STRESS TEST BY TIME



PLOT OF NUMBER OF TRANSACTIONS STARTED PER MINUTE BY TEST TIME
FIGURE 2.1.1

RESPONSE TIME

FOR FIRST CTR STRESS TEST BY TIME



PLOT OF THE AVERAGE RESPONSE TIME (SEC) BY TEST TIME
FIGURE 2.1.2

CPU TIME

CONSUMED IN PREPARING, PROOFREADING, AND PRINTING ONE DOCUMENT

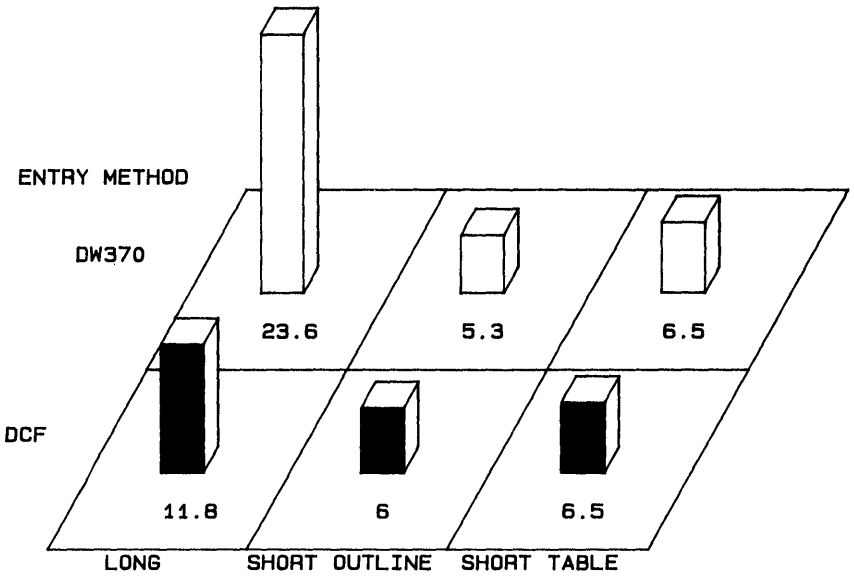


FIGURE 2.2.1

VIO COUNTS

REQUIRED FOR PREPARATION, PROOFREADING, AND PRINTING OF ONE DOCUMENT

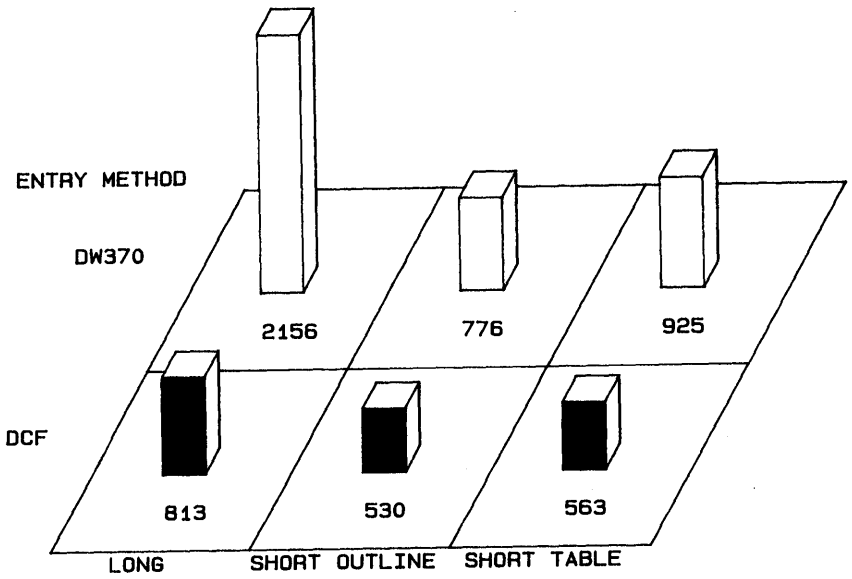


FIGURE 2.2.2

CPU TIME

CONSUMED IN PREPARING, PROOFREADING, AND PRINTING ONE FIVE PAGE DOCUMENT

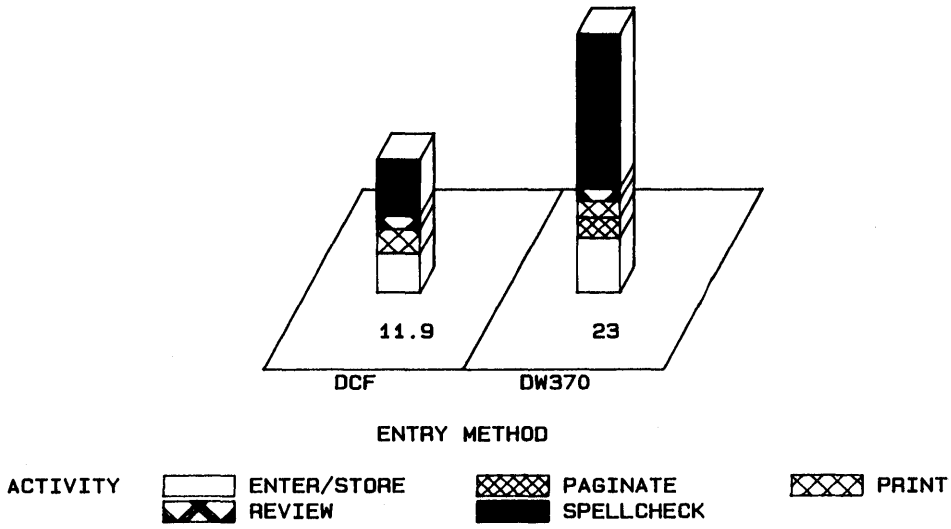


FIGURE 2.2.3

VIO COUNTS

REQUIRED IN PREPARING, PROOFREADING, AND PRINTING ONE FIVE PAGE DOCUMENT

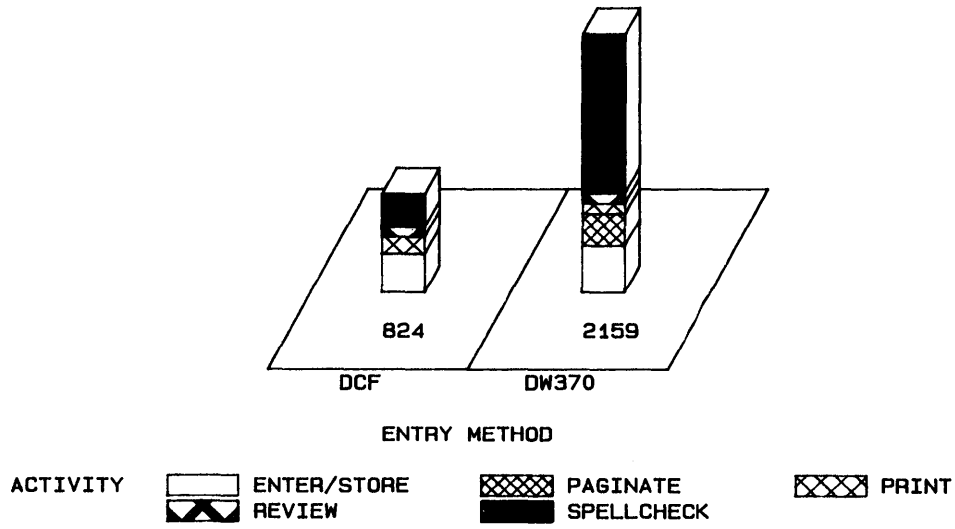


FIGURE 2.2.4

MINI-DISK SEEKS

MEASURED IN PREPARATION, PROOFREADING, AND PRINTING OF ONE FIVE PAGE DOCUMENT

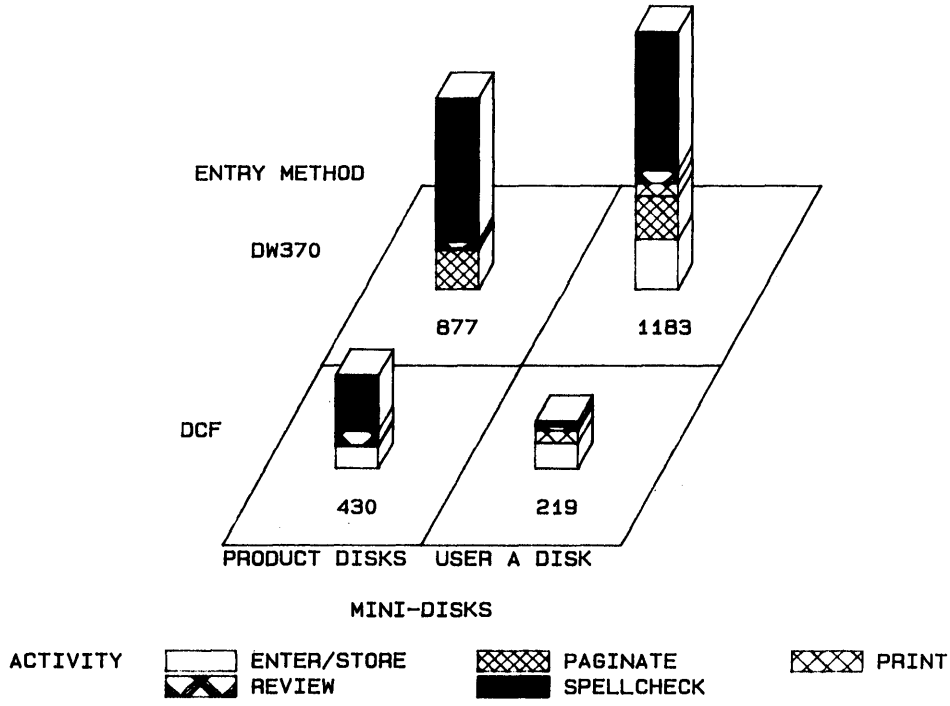


FIGURE 2.2.5

I/O Tuning in the MVS/XA Environment

Rich Olcott
Schering-Plough Corp
3022 Jackson
Memphis TN 38151
(901) 320-2867

After a brief consideration of tuning strategies and MVS/370, we explore the major performance-impacting innovations of the IBM XA hardware and MVS/XA software. We then apply the general principles to specific cases in locating and correcting performance problems arising in MVS/XA-managed DASD and tape subsystems.

The phrase "I/O tuning" covers a multitude of activities and concerns, ranging from planning the acquisition and configuration of hardware to the correction of design flaws in an application program. All of these center on the fact that waits for I/O completion are the largest source of delay in most computer systems. Speeding up user response time generally reduces to eliminating unnecessary I/O and then making sure that the remaining activity proceeds as expeditiously as possible. IBM's introduction of Extended Architecture (XA) hardware and software have changed the way I/O delay is measured, and thereby have enabled I/O tuners to adopt new methodologies for managing their systems.

Systems tuners have developed three general strategies for determining what changes should be made to improve performance. Explicitly or implicitly, most practitioners use all three at different times. For lack of better names, we will refer to them as bottom-up, top-down, and middle-out. The bottom-up strategy depends on "Rules of Thumb", sets of thresholds which on the average separate satisfactory and unsatisfactory systems. For instance, one time-honored rule of thumb is that no path element between CPU and device should be run at greater than 30% utilization. A performance analyst using the bottom-up strategy might generate an RMF/370 channel report, scan the "%busy" column, see no numbers larger than 30, and conclude that the system was healthy during the interval covered by the report.

That conclusion is based on several hidden but system-specific assumptions. The first has to do with the Rule of Thumb itself, the second has to do with the report, and the third has to do with the system the analyst is monitoring.

The 30% rule derives from the M/M/1 queueing equations [1]. These equations describe the steady state of a system with a single server. Their solution allows us to calculate how long an arriving customer will have to wait for that server to finish with a prior customer:

$$QT = ST * U / (1 - U)$$

where QT is the average queued wait, ST is the average service time, and U is the server utilization, the average fraction of its time spent busy with a customer. (This equation makes certain assumptions about the customer and server behavior. Those assumptions appear to have about equal degrees of validity for both MVS/370 and MVS/XA so we won't go into them here.) The behavior of QT/ST is shown as the solid line in Figure 1. At low utilizations waiting times are small. As U approaches 1 QT approaches infinity. Despite the appearance of the graph, there is no "knee of the curve", no special point where a sudden rise begins. The location of the apparent knee depends upon the vertical scale used in making the plot. On a graph with equal horizontal and vertical scales, the function has a 45° tangent at U = 0 and rises with an increasing acceleration from there. However, QT turns out to be exactly half of ST when U = 1/3. Thus, the U = 30% rule attempts to keep the queue wait below half of the service time. As we shall see, M/M/1 queueing does not necessarily apply to MVS/XA DASD or tape.

The RMF/370 channel report, like the RMF/370 device report, conceals another system-specific assumption. RMF/370 [2,3] is a sampling monitor, activated at regular intervals. At sampling time, the I/O data collection routine checks the device and I/O control blocks, noting the current state of each. U for a device is calculated as the ratio of "busy" samples to total samples taken during a reporting interval. The problem is that the calculated U may be biased if the sampling interval is at all synchronized with the activity being measured. Consider the school child who believes that Teacher lives at school, because Teacher is in the classroom whenever the child checks. In a system as complex as a running MVS/370, it is difficult to assay one way or the other whether such a bias exists.

The third assumption in the Rule of Thumb strategy is to assume that if the over-all system appears healthy, then all the applications are delivering acceptable performance. When it is put so bluntly, the risk in this assumption is immediately apparent. It is easy to write a scenario in which, for instance, a production PROCLIB and a CICS-active ISAM file are on the same DASD volume as a page data set. When a CICS region encounters a page fault, all of its activity ceases until the fault is resolved. If fault resolution must wait until completion of a long PROCLIB directory search or an ISAM overflow area update, then many CICS users will experience poor response time even if the DASD volume in question delivers nominal performance.

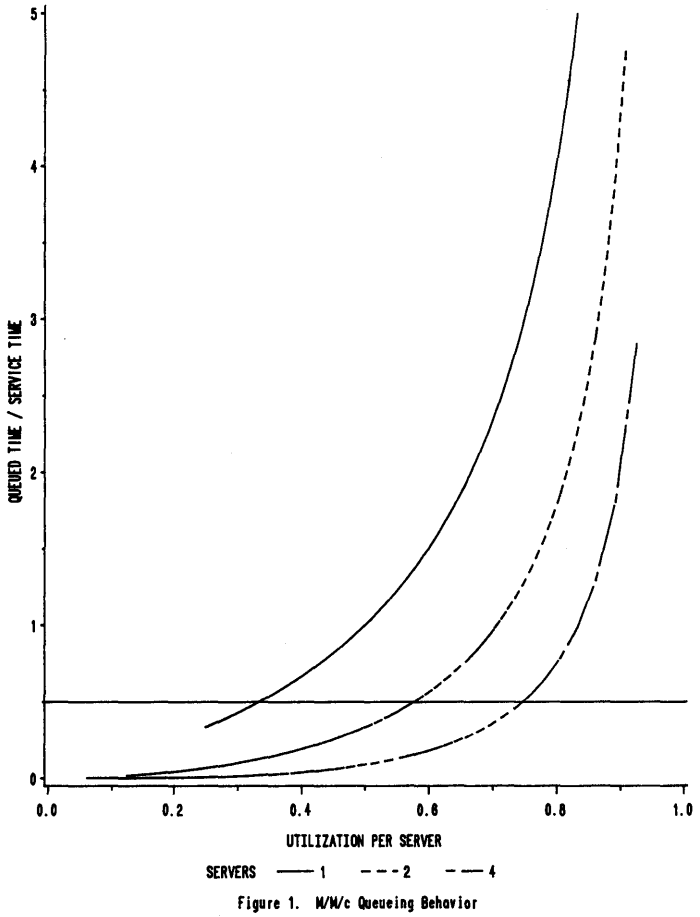


Figure 1. M/M/c Queueing Behavior

The bottom-up strategy concentrates on the device statistics: a performance problem is defined to exist only when some device-level Rule of Thumb is being violated. In contrast, the top-down strategy defines a performance problem in terms of objectively measured service level goals: "Is the 90th percentile TSO trivial transaction response time less than our published threshold?" If so, there are other projects to work on. The performance analyst starts studying the I/O subsystem only if some component of the installation's workload is not fulfilling its goals, and then only if measurement shows that I/O is the major source of delay.

The middle-out strategy attempts to finesse performance problems before they occur. The heart of this strategy is a model of the I/O subsystem, built from known device characteristics and expected workload levels. The model is used to calculate expected service delivery at the application level. If the modeled service levels do not meet the installation's service goals, the I/O subsystem is re-configured on paper (or silicon) until goals are met, at which point the modeled subsystem is translated to real devices and cable connections. The model's use is not over, however. One of its by-products is a set of expected device performance statistics. These statistics then become installation-specific Rules of Thumb. Significant deviations between measured and modeled device and path element activities are danger signals, to be investigated and addressed before application-level performance problems appear.

A strategy is no better than its implementation and the quality of the supporting quantitative data. "You can't manage what you can't measure", and conversely, how you manage is determined by what your measurements represent. The introduction of XA hardware and MVS/XA software have changed the implementation of all three strategies by changing the information upon which they depend.

XA Hardware Innovations

The most dramatic innovation in XA hardware [4] is undoubtedly the 2 GB virtual address space. The Dynamic Address Translation facility ("DAT box") transforms a virtual address to a pointer referencing a real main storage location. An XA DAT box can handle a maximum virtual address 31 bits wide, giving an addressable range 128 times larger than the S/370 16 MB maximum. However, this change's impact on the I/O subsystem has been minimal, reflected mostly in an increased load on the paging volumes.

The other major XA innovation [5] has had a much greater effect. S/370 I/O is managed from the CPU. Channels are directly attached (logically and physically) to the central processor. CPU time is spent choosing a path for each I/O request, adjudicating contention for path elements, monitoring I/O request progress, and recovering from errors. The XA architecture places a dedicated subsystem between the I/O equipment and the CPU. The internal structure and nomenclature of this channel subsystem (CSS) vary with the hardware. The CSS contains one or more microprocessors and serves as the connection point for all I/O cabling. Microcode running in the microprocessor(s) has taken on all of the above responsibilities for I/O management. The host operating system need only pass an I/O request to the CSS and then accept an interrupt after the CSS has completed the indicated transfer to or from main memory.

The most important implication of this architectural change appears when we look at contention in the I/O subsystem. Consider the cabling diagram in Figure 2. Looking downstream, the CPU or CSS communicates through a channel [*] to a DASD control unit (CU), which can manage up to four strings of DASD devices. Each string contains up to 8 actuators and is under direct control of a head-of-string (HOS). Looking upstream, the HOS contains a string-switching feature so that connections can be made to two CUs. Similarly, the CU has a channel-switching feature that allows it to connect to up to four channels.

Contention arises because a transmission must gain exclusive control over an entire path (channel, CU, HOS) between CPU (or CSS) and device. None of the path elements is able to support transmission of multiple control or data streams, in S/370 or XA architectures. Suppose a system needs to read a block from DASD. For the moment we simply assume that the resident operating system has already ensured that the device is not already busy. The I/O's lifetime begins with the outbound transmission of positioning information to the device. If any intermediate path element (CU or HOS) is busy with another I/O operation, the attempted transmission is thwarted. Our read request must be queued for retry at a later time.

In fact, the exposure to path contention occurs at least twice for each DASD I/O request. Because of the channel contention problem, all current S/370 and XA rotating-media devices support a protocol called Rotational Position Sensing (RPS). RPS allows the device to disconnect from the channel during periods of mechanical delay, thus freeing at least some of the path elements for other work. No freedom comes free, however. When the device is ready to transmit the designated block, it must gain control of a path back to the central system. If some intermediate path element has become busy on another I/O during the interval since disconnect, the reconnect fails. Again the request is queued for retry, but in this case the retry cannot occur until the platter completes a full rotation and is again positioned properly to read the desired block of data. We have just witnessed an RPS miss.

Herein lies the I/O advantage of XA over S/370. Recall that S/370 channels report directly to the CPU. S/370 software identifies each outstanding I/O with the channel upon which it started. S/370 architecture requires that all the subsequent activity for that I/O (initial responding interrupt, any necessary redrive activity, initial data transmission, and any later command and/or data transmission) take place via the same channel. S/370 upstream I/O thus fits the definition of a single-server queueing system, no matter how many channels physically connect the device to the CPU.

XA's introduction of the CSS changes that picture. The CSS is able to manage I/Os at the device level (see below). A reconnect can exploit any available path from the device back to the CSS, not just the path used to initiate the I/O. In terms of queueing theory, the reconnect follows M/M/c statistics (where c is the number of servers). In an M/M/c system,

$$QT = ST * C(c,U) / (c - U)$$

where C(c,U) is Erlang's formula for the probability that a new customer will find that all c servers are busy if the average server is U/c percent busy. The dashed lines in Figure 1 show

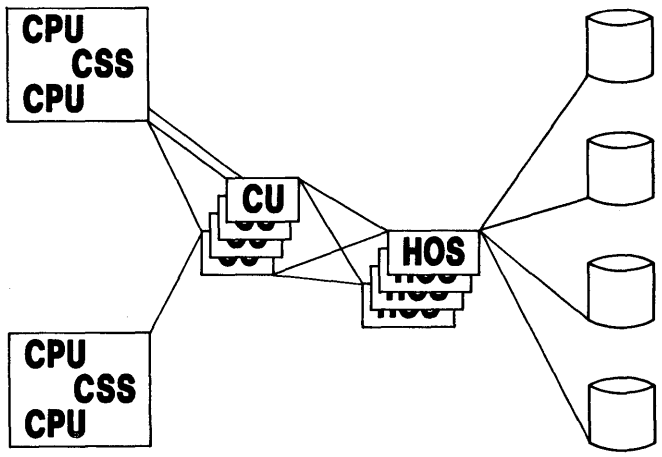


Figure 2. Typical DASD Cabling Diagram

---

* - IBM uses "channel" to refer to the S/370 physical and logical connections between CPU and control unit, and "channel path" to refer to the XA physical connection alone. We will use "channel" for both architectures, at the risk of losing this subtle distiction.

the behavior of QT for 2 and 4 servers. It is clear that XA's Dynamic Path Reconnect (DPR) capability has significantly eased constraints on path element busy for DASD. The XA equivalent to the S/370 30% path element Rule of Thumb can be as high as 70% for equivalent performance. Note well that the device itself is a single server in both architectures. The 30% device Rule of Thumb is unchanged.

Another important implication of the CSS architecture has to do with I/O measurement. Recall that RMF/370 is a sampling monitor, counting I/O requests but estimating time spent satisfying them. The I/O-dedicated resources in the CSS can measure delays and active times directly. For an analogy, consider measuring airplane in-flight time. The S/370 method would start with more-or-less regular satellite photos of a sample region. Count planes in the air in each picture. The estimated flight time is is the total number of in-air images, times the duration of the sampling interval, divided by the total number of flights as determined by a separate census. The XA equivalent would be to have each airport's control tower log the take-off and arrival time of each flight.

The XA I/O management subsystem is organized around a set of control blocks that interface mainframe software and CSS microcode. Each I/O request is represented by an Operation Request Block (ORB), and each interrupt by an Interruption Response Block (IRB). The subchannel is the CSS microcode's image of the device. Each unique device is represented by a Subchannel Information Block (SCHIB). The SCHIB contains, among other things, a list of paths that service this device, an indicator if timings are to be kept for this device, and a pointer to the device's Channel Measurement Block (CMB).

We need one more definition. Recall that the S/370 channel is the highest bottleneck in its I/O architecture. If the channel is busy, than the S/370 I/O request must wait until the channel is free. The most efficient way of managing the wait is to construct a FIFO queue of I/O requests against each channel. The mainframe software does this by maintaining what in MVS/370 is termed the Logical Channel Queue. XA has no channel per se, or rather has so many that they are rarely the bottleneck. The new heir to that throne is the CU: if the CU is busy than an XA I/O request must wait until the CU is free. Again the solution is a FIFO queue, this time headed by the set of control units that can connect to a device, or better yet, a common set of devices. We have just described the Logical Control Unit (LCU): the queue formed in the CSS to represent each set of devices that shares a common set of physical control units.

We can best understand subchannel operation by following an I/O request [6]. Activity at this level starts when the mainframe software issues a Start Subchannel (SSCH) instruction whose operand is a pointer to the SCHIB for, say, device 5A2. The subchannel (really, the combination of general microcode routines and the specific SCHIB) starts a timer and places the ORB on a queue of requests destined for 5A2's LCU. This first timer measures the Pending time, the delay until the request gains access to a specific channel path. At that point, the Pending time is recorded in 5A2's CMB and a second timer is started. This timer measures the Connect time, during which control information and data pass along the channel path. The Connect timer also runs during error correction activity. For some devices, function may allow or require the device to relinquish path control during the period that the ORB is being serviced. Whenever that occurs, the Connect timer is stopped and a third timer, for Disconnect time, is started. When path control is regained, Disconnect is stopped and Connect is restarted.

Activity continues until 5A2 signals Device End, signifying the completion or suspension of the ORB-specified operation. The subchannel stores the accumulated Connect and Disconnect time totals in 5A2's CMB, fills in an IRB, and passes an interrupt back to the mainframe software. Some processor models go so far as to record separate total delays for each level of path element. In principle, then, the software is aware of the exact amount of time that each I/O spent waiting for a path, transferring control or problem data, and holding a device but not doing data transfer.

There are some data integrity problems in the CMB interface. Each of the accumulators (Pending, Connect, and Disconnect) can overflow during the servicing of a given I/O. The overflow threshold varies with the processor class; the limit for 308x equipment, for instance, is 8.3 seconds during any single channel program. However, the hardware system only detects and reports on Connect and Disconnect overflows; reported Pending time measurements should always be treated as lower limits on the true value.

MVS/XA Software Innovations

All of the aforementioned activity takes place at the hardware and microcode levels, available to any program or operating system running on an XA processor. How does MVS/XA in particular exploit the XA CSS architecture and measurement facilities?

We have already described the advantage obtained from DPR, which reduces the likelihood of DASD RPS miss by increasing the number of paths from the device back to the host. We have also alluded to the potential advantage to be gained from offloading much of IOS's processing load to the CSS. We should note the relief of a constraint to I/O subsystem growth. The S/370 architecture defined the device address as a one-hex-digit channel-id followed by a two-hex-digit sequential unit number. In effect, this limited the S/370 I/O subsystem to 16 addressable channels, each with a maximum of 256 units. The XA architecture has removed the channel-device ratio constraint entirely and its almost-4-hex-digit device number in principle allows a maximum of 32K devices.

XA's direct measurement of I/O timing has changed the complexion of MVS/XA system measurement tools and has altered strategies for using them. The major art in using MVS/370's tools was in tracing state percentages and EXCP (EXecute Channel Program) counts back to delay sources. In MVS/XA, we can look at the delays themselves, because each timing measurement corresponds to a unique set of delay causes. We will return to this point for specific device types after considering the MVS/XA tools themselves.

The heart of the MVS/XA I/O measurement subsystem is the CMB, the interface between the CSS microcode and the mainframe operating system. The CMB's are constructed by the System Resource Manager (SRM) at IPL time. SRM automatically builds enough CMBs to account for all DASD and tape devices declared in the active IOCDS. If the analyst wishes to track times for other device classes, they must be requested using the CMB= parameter in the IPL-time IEASYSxx PARMLIB member. The device(s) of interest must be attached to a block channel: byte multiplexor channels are not equipped to measure or report device time data.

The System Management Facility (SMF) [7] software provides data on resource utilization by each batch job, TSO user, and started task. Its complete XA benefits are only available for workloads subject to Type 30 recording, and then only for devices for which CSS measurement is active. The Type 30 record for a step contains connection time data at two levels. At the lower level, the record contains one EXCP section for each unique device allocated to a DDNAME, whether by JCL or dynamically. The section contains several fields that relate to that device, including the IOGEN/MVSCP-derived device number and the allocated DDNAME. If multiple datasets are allocated under a common DDNAME to a single device, the quantitative data are consolidated into a single EXCP section. Three fields describe the I/O that took place to the device:

- SMF30BLK - the number of blocks transferred to/from the device
- SMF30BSZ - the largest blocksize observed for the dataset
- SMF30DCT - the device Connect time for that I/O

Much of performance management consists of searching for anomalies, and here is our first opportunity. The product of SMF30BSZ and SMF30BLK is an upper limit on the number of bytes transferred, equal to the byte count for fixed and VSAM record formats. The product of Connect time and channel speed should also be equal to the number of bytes transferred. If the latter measure is significantly larger than the former, we may have an indication of serious inefficiency in the I/O for that dataset. We will return to this point below.

Field SMF30TCN in the I/O Activity section contains the total device Connect time for all I/O (except paging I/O) done on behalf of this step. The difference between SMF30TCN and the sum over SMF30DCT thus represents I/O to datasets that are not explicitly device-allocated: linklist libraries, security dataset, catalogs, JES spool, etc. It also includes I/O done by OPEN/CLOSE (for instance, to VTOCs or tape labels) that does not result in transfer of dataset content. Note that neither paging I/O nor VIO is included in SMF30TCN or SMF30DCT totals: the corresponding physical I/O is done by the Auxiliary Storage Manager and is not charged to individual address spaces.

There is another, very important datum in the Type 30 I/O Activity section. Bit 0 of field SMF30MFL is set to 1 if channel measurement was disabled for any I/O (DD or system level) in this step. There is no indication as to which individual statistic is in error. Thus, any records with this bit set should be rejected in preparing an I/O report.

Resource Measurement Facility (RMF) is the other major IBM-supplied MVS performance tool. Originally written to feed performance data to the SRM, it is the central source for system-level utilization and delay data. Other vendors have produced similar monitors, all of which use the same underlying hardware facilities and produce similar sets of data elements. They compete on the output side, vying for the analyst's favor on the basis of summarization strategy, ease of use, and presentation formats. We will describe only the IBM product here, but the analyst would be well-advised to consider the market before settling on any particular product.

There are actually three different RMF monitors that can operate in a running MVS/XA system. Monitor I, still closely coupled to the needs of SRM, generates SMF Type 70-77 records as a by-product. The records are generated periodically and contain summaries (usually totals) of system resource statistics. Record Types 73 (Channel Activity) and 74 (Device Activity) are the most relevant to our discussion here. They are used by the RMF Post Processor to inform the corresponding RMF reports. They can also be reduced by other programs to generate custom reports or load a database for ad hoc queries.

Monitor II is used for close scrutiny: traces of specific data elements for short periods, snap-shot reports of system status, and on-line queries. Monitor II is an offshoot of the Monitor I facility, and there is a good deal of overlap in their orientation and the data elements they present. In particular, both Monitor I and Monitor II track device activity in terms of I/O rates and mean Pending, Connect and Disconnect times. Furthermore, both support an additional statistic that cannot be measured by the CSS: I/O queueing time within the mainframe.

We have said that Pending time represents path wait. However, that wait only begins when MVS's I/O Supervisor (IOS) passes the ORB to the CSS. An I/O request's life cycle passes through several stages before CSS ever sees it. The application program first issues, say, a GET request against a particular dataset. The GET request is directed to Access Method (AM) software which builds a channel program designed to satisfy the request. The AM uses EXCP (or an equivalent call) to pass the channel program to IOS. IOS will connect the channel program to an ORB and notify CSS, but only after issuing a Test Channel (TCH) command and checking the device's Unit Control Block (UCB) to see whether or not the channel or device is already busy to this processor. There is nothing to be gained by overloading the CSS queues. If IOS knows that the request will encounter a busy condition downstream, IOS queues the request in the mainframe until the blockage is cleared. This additional delay, IOS Queue time, is measured by sampling the IOS queue length and then calculating from Little's Law:

$$IOSQ = \frac{\text{(Mean sampled queue length)}}{\text{(I/O rate)}}$$

Monitor III adopts a different perspective on the system workload. The philosophy implicit in Monitors I and II is to concentrate on the system's resources, diagnosing delays by asking which resource is overcommitted and then which tasks are competing for that resource. Monitor III concentrates on the workload itself. The monitor periodically samples the state of each task in the system. If the task is in a wait state, the monitor records what the task is waiting for. The data are summarized once each reporting interval (commonly either 60 or 100 seconds). The entire matrix (task by delay source by report time) is maintained in virtual storage for online reference. Monitor III's presentation side is designed to offer quick answers to fire-fighting questions like, "What workload is suffering the greatest delay?" and, "Which tasks are competing with job BVSD4000 for DASD volume MVSA01?"

Before leaving the software tools area, we should mention several measurements that are not XA-specific but are no less important in the XA environment. The EXCP count is a still vital datum. Technically, EXCP is but one of several I/O drivers supported by IOS. ASM, VSAM, VTAM, OLTEP, JES3, and program fetch all have their own specially-tailored interfaces. Nevertheless, each invocation of IOS is referred to as an "EXCP".

Because I/O quantification in earlier versions of OS was limited to EXCP counts, a "one EXCP, one block" interpretation has arisen. That interpretation is not necessarily correct and has generated some confusion. Channel programs can be of arbitrary complexity: one EXCP can result in the transfer of one, several, or no blocks of data. Recent versions of IOS have attempted, unsuccessfully, to clarify the situation. For the sequential access methods (QSAM, BSAM, BPAM), true block counts are reported as "EXCP counts" and true EXCP counts are not reported at all. This explains why attempts to tune an application by revising the channel programming (specifying an increased buffer count, for instance) seem to have no effect. There may indeed be a reduction in IOS invocations, but the transferred block count remains the same.

The EXCP ambiguity is the reason that one should use the "Connect time per EXCP" ratio cautiously. The ratio may be high because of very "expensive" I/O, but it may also be biased in either direction by non-transfer EXCPs. High Connect time may be due to error correction activity for the device or any other device-specific activity that elicits no data transfer but yet does not release the path. No-data EXCPs may be used to release a RESERVE, force a rewind, or carry out some other device management task that must be directed from the mainframe.

The other important non-XA I/O metrics are network-oriented. Unfortunately, network measurement is an even less mature discipline than mainframe device measurement. The ideal metric is one that can be economically and reliably evaluated, easily understood, and readily compared to some straightforward measure of capacity. None exists.

Part of the problem is that "the connection" is complex, possibly comprising communications and terminal controllers, protocol convertors, multiplexors and modems as well as wires and microwave beams. Each component of "the connection" is a potential bottleneck. Part of the problem is that "the message" itself has different sizes when viewed by different levels of the system. Each layer may use compression techniques to reduce data length, may slice the message into smaller packets or merge multiple packets into longer transmission units, and may add header and trailer

information for message routing and identification. Part of the problem is that network connections have much poorer reliability than do local connections. Message integrity requirements have forced protocol designers to work with short transmission blocks and several layers of redundancy, all of which reduce effective connection capacity. Part of the problem is that network management itself requires network time for its function. All network protocols include provision for control transmissions before and after data transmissions.

Part of the problem is historical. "Growth at need" of existing networks have led to a variety of protocols and message structuring techniques used even within a single network discipline. Consider the range of "LUs" available under VTAM, for instance, and ponder on the measurement consequences of a gateway between a bit-stream system like SDLC and a packet-oriented system like X.25. Part of the problem is cultural. Computer communications networks are at the interface between two industries that each have developed tightly reinforced ideas of "the right way" to do things. Part of the problem is instrumental. Even given satisfactory metrics for message size and transmission channel capacity, there is no current technology for accurate measurement of transmission time.

XA itself has no solution for the problems. Many of the phenomena that affect network capacity are beyond the purview of any single CSS or set of CSSs. My purposes in raising the point are to warn the new XA user that networks are still untamed, and to encourage those who are seeking solutions.

System-level I/O Tuning Facilities

Before turning to the treatment of specific device types, we should note several I/O-related system-level parameters that are either new or enhanced in MVS/XA.

I/O in S/370 (classic and XA) is an asynchronous process. The CPU passes a channel program to the I/O subsystem and then takes up other business. The subsystem posts an interrupt to the CPU when the I/O requires the CPU's participation. Any deviation from a one-to-one SSCH-to-interrupt relationship is an error. A waiting I/O that does not receive a timely response is said to be suffering a missing interrupt. A device that generates repetitive unsolicited interrupts is displaying a hot I/O. Both types of errors have performance impacts. A missing interrupt stops the application that is waiting for the I/O in question. Responding to the repeated challenge of a hot I/O may so dominate the CPU that no other work is accomplished.

IOS contains a subsystem, the Missing Interrupt Handler (MIH), dedicated to detecting and coping with both types of error. MVS/XA's MIH covers a wider variety of phenomena that does MVS/370's. Furthermore, MVS/XA's designers have allowed the system tuner to specify parameters that control MIH's vigilance. After all, some device types have inherently slower responses than others. Why spend resources recovering from a missing interrupt that is tardy but not AWOL?

The parameters may be placed in a new PARMLIB member, IECIOSxx [9], selected by an IOS=xx start-up parameter. For missing interrupts, one can specify device-specific surveillance intervals. If the device is pending at both the start and end of an interval, and its I/O count has not changed, then a missing interrupt is diagnosed and MIH logs the event and attempts to recover, for instance, by redriving the attempted I/O. For hot I/O's, the system programmer is invited to specify how many unsolicited interrupts over what interval are to be considered "hot" for each general device type, and what automatic actions should be taken in response to the first and later misbehaviors.

There is an interaction between the MIH expiry time and the accuracy of reported Pending times. SRM has taken advantage of the fact that there are two timers measuring I/O delays. The MIH timer runs in the mainframe and senses when any particular I/O has lingered beyond the limits prescribed in IECIOSxx. The Pending timer runs in the CSS and can overflow. Because the MIH subsystem inspects each the state of each device once each MIH interval, SRM treats the MIH timer interval as an effective maximum Pending time. The RMF/XA-reported Pending times have been adjusted to include MIH-sensed outages, but note that the resolution of that adjustment is limited by the MIH interval for the device in question.

I/O load balancing operates somewhat differently in MVS/XA, but retains the same basic concepts as the MVS/370 version. SRM has three load balancers, for I/O, CPU, and real storage. Each attempts to ensure that its particular resources are properly loaded. Early versions of the I/O balancer simply considered the overall usage of the I/O subsystem. Later versions keep separate accounts for subsets of the total activity. MVS/370's load balancer subsets by logical channel, defined as the set of devices that share a common set of physical channels. The busy-ness for a logical channel is measured by the fraction of its I/O requests that have been deferred because of physical channel busy. If any logical channel %busy exceeds an installation-specified system-wide threshold, then the load balancer looks for the heavy users of the devices on that logical channel. The load balancer then recommends that SRM swap-out that user or users.

MVS/XA's I/O load balancer [8, chapter 5] is very similar. Of course, "channel" is a non-word in XA so one speaks instead of physical and logical channel paths. From the tuner's point of view, SRM's "logical path" is identical to IOS's "logical control unit". MVS/XA logical path %busy is calculated from measured connect times rather than request deferrals. The truly novel element in MVS/XA's I/O load balancer is separate treatment of logical paths serving tape, DASD with dynamic reconnect capability, and DASD without dynamic reconnect. MVS/XA has recognized that the performance of the three device classes respond differently to a given level of channel utilization. Each class of logical paths has its own algorithm for calculating path %busy and its own overload threshold. The algorithm parameters are set out as a new set of keywords in the active IEAOPTxx member of PARMLIB.

Selective enablement is SRM's subtlest response to the innovations in XA architecture. The placing of CSS between CPU and peripherals broke the bond between I/O request and channel and thereby led to DPR. It also broke the bond between I/O request and CPU. There is no longer a requirement that the interrupt be serviced by the CPU that issued the I/O request. Consider the activity that must occur when an I/O interrupt is presented to a CPU running problem code. First, the status of the currently running task must be saved. The current contents of the high-speed cache(s) and the instruction pipeline must be flushed. The high-speed cache must be loaded with the code to be executed in response to the interrupt. The new instruction stream must be started through the pipeline. Machine cycles are stolen while the CSS accesses real storage for data transfer. First-level interrupt handling (FLIH) code and possibly some other routines must run before interrupt disabling can be released and control returned to the dispatcher. Meanwhile, some of the first application's favorite virtual storage may be paged out before the application regains control of the CPU.

Now consider the benefit to be gained in a multi-processing complex if one CPU services all the I/O interrupts. On the one hand, the I/O server spends a higher fraction of its time in IOS code and therefore is less liable to context switching. On the other hand, the other CPUs are also less burdened by context switching so a higher fraction of their time is spent in problem state. Everybody wins.

The situation as described would dedicate 50% of a dyadic processor to interrupt servicing, clearly an untenable strategy. The XA SRM provides a dynamic control instead, based on IOS's use of the Test Pending Interupt (TPI) instruction. IOS plays the odds by issuing a TPI just prior to

the end of its FLIH routine. If another I/O interrupt is waiting, IOS can return to the top of FLIH and service that interrupt with no context switch. The SRM selective enablement algorithm responds to the fraction of all interrupts that are serviced by this TPI tactic. If the fraction is too low, then too many processors are doing I/O servicing; SRM refuses to pass any more I/O interrupts to one of those processors. If the fraction is too high, then too many interrupts are kept waiting for service; SRM makes another processor eligible for I/O interrupt activity. The installation defines "too low" and "too high" in the CPENABLE= parameter of the active IEAOPTxx PARMLIB member.

Application to DASD

DASD I/O management is the most mature CPE discipline. The waits within the DASD subsystem are well understood, instrumentation is in place and dependable, and the literature [10] and the marketplace provide bridges between symptom and treatment.

Refer again to Figure 2 and recall the operational definitions of:

IOSQ time - delay waiting for control of the UCB.
Pending time - delay in the CSS waiting for control of the path.
Connect time - time spent transferring control information or data, or error correction time.
Disconnect time - time during which the device is dedicated to an I/O request, but not connected to the CSS.

How is the tuner to translate reported wait times into recommendations for action? Let us first describe each in terms of specific DASD device and path activities. We can then turn to tuning strategies.

First a rapid review of DASD operation in S/370. A simple DASD channel program might be made up of the command sequence "SEEK, SET SECTOR, READ". For 3350 and earlier technologies, each of these commands must be transmitted separately. The CSS-to-device link is connected for the SEEK and then disconnected. The DASD actuator moves to the requested track. The link is reconnected for SEEK success acknowledgement and SET SECTOR transmission and then disconnected once again. When the platter has rotated to the proper sector, the link is reconnected yet again for READ command trans-mission and data transfer, followed by a final disconnect. For 3380 (and, one supposes, later) devices, the SET SECTOR command can piggyback on the SEEK transmission, thus avoiding a reconnect delay exposure.

The low-level logic in the hardware includes substantial error detection and correction capability. If an uncorrectable error is detected in any step of the I/O cycle, the CSS connects to the device and steps through an extended retry procedure until either an error-free transmission is completed or a repetition threshold is exceeded. There is a sustained connection to the device throughout this procedure, so Connect time is accumulated and the path is unavailable to other transmissions until it completes.

DASD Connect time is thus made up of protocol time for each connect and reconnect, command transmission time, data transmission time, and error correction time. Connect time is more important as a performance determinant of paths than devices. For a single DASD I/O operation, the Connect time is often less than 10% of the response time. However, a higher path element's utilization is the sum of the Connect times for all transmissions through it. A CU managing 8 devices, each of which is connected 10% of the time, is itself 80% busy.

DASD Disconnect time is mainly attributable to reconnect path delay and waits for mechanical events (SEEK completion and platter rotation). As we have seen, DPR has essentially eliminated reconnect path delay. For reasonable path utilizations, reconnection waits average 10% or less of the delay for each of the mechanical waits.

DASD Pending time is pure path contention. An I/O starts accumulating Pending time when the CSS first accepts the corresponding ORB. Pending time stops accumulating when the first connection to the device is accomplished. Along the way, the request may have to wait for access to the channel path, to the CU, and to the HOS. Each looks like a M/M/1 server, extending the response time by a factor of [U / (1 - U)], where U is the fractional utilization of the corresponding path element. The U for the channel path and CU are directly calculable as

$$U = \frac{[\text{SUM (Connect times)}]}{\text{(Measurement duration)}}$$

where the summation is over all I/O using that path element during the measurement interval. HOS utilization is a tougher nut. Recall that an S/370 CU may contain a channel switch that allows upstream connection through up to 4 channel paths. There is no requirement that all four paths lead to the same system. If in fact they connect to different systems then the CU, the HOS, and the devices behind it are said to be "shared" among those systems.

Sharing DASD leads to some problems in system measurement and data management [11]. One must be especially wary of write integrity exposures. Suppose, for instance, that a device is shared between two systems. TSO users in each system are running a common application that updates records in a file on that volume. If both users happen to access the same block of data, the second one to rewrite the block will overlay the updates made by the other user.

S/370 and XA architectures address this exposure with the RESERVE and RELEASE channel commands. When a system issues the RESERVE command against a device, all other sharing systems sense that device to be busy at HOS. That condition persists until the holding system issues the RELEASE command. In MVS/XA, Pending time includes delays waiting for a device RESERVEd by another system. HOS U must thus include both Connect time (from all sharing systems) and that portion of Pending time that is actually RESERVE-but-idle time. Unfortunately, the latter is not directly measurable. The closest equivalent is a field in the SMF Type 74 record, SMF74UTL. This field reports "dog in the manger"-hood: the sample count for which the reporting system had RESERVEd the device but had no outstanding I/O against it. The analyst has the task of calculating

$$\text{HOS } U = \text{SUM}\left(\frac{\text{SUM (Connect times)}}{\text{(Measurement duration)}}\right) + \text{SUM (SUM [(SMF74UTL) / (SMF74SAM)])}$$

where the inner SUM is over all devices on the string, and the outer SUM is over all RMFs reporting on the string. Of course, U for a shared CU must also be calculated by double sums, but that expression does not require the terms in SMF74UTL.

IOSQ time is peculiar to sharable devices. A DASD device is sharable because it can service one application's requests independently of another's. However, that sharing is really serial control, not concurrency. The actuator arm cannot move in two directions at once. IOS recognizes this limitation by queueing I/O requests against a device or path it knows to be busy, giving rise to IOSQ time. Hence, IOSQ time represents competition within a single system for a given device or path. The analyst can easily diagnose the difference from the IOSQ times alone. If many devices on a path show high IOSQ time per SSCH, it's a path problem; if one device does, it's that device's problem.

Armed with an understanding of the meaning of the reported times in the DASD context, we can turn to their application in tuning. The bottom-up strategy, remember, starts with the assumption that if all the device statistics look good ("happy numbers"), then the system is probably working well. What RMF "happy numbers" apply to DASD in an XA environment? How should one approach an RMF Direct Access Device Activity report?

Dev Num	Volume Serial	LCU	Device Activity Rate	Avg Resp Time	Avg IOSQ Time	Avg Pend Time	Avg Disc Time	Avg Conn Time	% Dev Conn	% Dev Util	% Dev Resv	Avg DS Open
860	PERM84	19	2.168	24	2	2.5	17.0	2.5	0.55	4.12	0.0	47.1
861	SYS001	19	8.882	39	12	1.3	18.9	6.9	6.15	19.74	0.0	34.7
862	PERM86	19	1.729	28	1	1.1	19.1	6.4	1.11	4.33	0.0	15.9
863	PERM87	19	1.727	24	0	1.2	15.2	7.2	1.23	3.77	0.0	8.6
864	A80011	19	9.930	28	6	1.0	17.1	3.9	3.84	20.34	0.0	26.3
865	A80009	19	5.638	24	2	1.2	18.0	2.9	1.61	11.47	0.0	13.8
866	PERM90	19	0.787	23	0	1.5	16.1	5.4	0.43	1.66	0.0	13.4
867	LIB100	19	2.020	40	3	0.6	15.6	20.4	4.13	7.27	0.0	14.6
	LCU	19	32.881	30	6	1.2	17.62	5.8	2.38	9.09	0.0	174

Figure 3. Direct Access Device Activity Report

There are at least as many ways to read an RMF report as there are people willing to tell about it. Some authors [12, 13] have developed extensive lists of threshold numbers for different devices and situations. I prefer to scan the report as a whole, simply looking for numbers that are out of line with their neighbors. After all, the odds are overwhelmingly against an entire DASD farm being uniformly out of tune. Usually the bottleneck volume or string will advertise its presence by numbers that stick out like a sore thumb. The tuning problem then reduces to understanding the behavior of that volume or string and then taking appropriate action.

For example, consider the partial report in Figure 3. The figures worth noting are the IOSQ time of SYS001, the Connect time of LIB100, and the Pending times of the entire 860 string (compared to the farm average, 0.4 msec). The Disconnect times are also a bit high for 3380 volumes. As it happens, SYS001 contains a very popular catalog and LIB100 is mostly batch program and PROC libraries. The SYS001 IOSQ time represents application and TSO allocation activity; the tuner's recommendation was to split the catalog across two volumes.

Long connect times on LIB100 are the result of extended SEARCH KEY EQUAL operations against PDS directories, combined with FETCH's use of chained scheduling to transfer a multi-block load module with a single EXCP. The question is, do these long Connect times lead to a performance problem elsewhere in this system?

Apparently they do. Notice that LIB100 is on the string with the long Pending and Disconnect times. LIB100's long connections have so tied up the HOS that I/O to the other volumes is being impeded (the "skew effect"). The effect is not large, yet, but could grow. That observation leads to a recommendation that the datasets on LIB100 be spread over several strings.

The top-down strategy [14] views performance from the application side. Figure 4a shows a summary of SMF data averaged over several runs of the SQRC jobstream. SUB_WAIT is the interval between the end of one job in the stream and the submission of the next. INITWAIT is the delay between submission and acceptance by a JES initiator. ENQ_WAIT is the delay due to ENQ interference. ALC_TIME is time spent in allocation, either tape drive assignment or DFHSM dataset recall. CPU_TIME is the total TCB and SRB time. DASDTIME and TAPETIME are the total durations of I/O to the named device types, estimated by EXCP count multiplied by system-wide DASD and tape response times. (XA Type 30 recording provides Connect times but not response times.) ???_TIME is the difference between the observed execution time and the sum of ENQ_WAIT, ALC_TIME, CPU_TIME (times an average MPL for that time-period), DASDTIME, and TAPETIME.

Jobname	SUB_WAIT	INITWAIT	ENQ_WAIT	ALC_TIME	CPU_TIME	DASDTIME	TAPETIME	???_TIME
SQRC001M	.	0:02	4:03	0:04	0:19	1:42	0:00	0:18
SQRC005M	0:48	0:03	0:00	5:14	0:36	7:39	2:38	-0:08
SQRC011M	0:33	0:01	0:00	0:05	2:31	0:52	0:00	0:22
SQRC012M	0:26	0:02	0:00	0:23	0:52	11:54	0:00	0:28
SQRC018M	.	0:02	0:00	0:43	0:18	0:47	0:00	0:13
SQRC024M	117:02	0:02	0:00	0:05	2:52	247:11	0:00	1:18
SQRC029M	0:44	0:03	0:00	0:19	1:53	2:08	0:00	-0:12

Figure 4a. Jobstream Timing Statistics

Stepname	DDname	Devtype	VOLSER	EXCPS	DSNAME
SQRC0241	PC1274CL	3380	A81003	155,364	SQRC.PC1274CL.DATA
SQRC0241	PC1274CL	3380	A81003	204,426	SQRC.PC1274CL.INDEX
SQRC0241	PC1275CL	3350	A50005	7,293	SQRC.PC1275CL.DATA
SQRC0241	PI0124	3420	027615	5,729	SQRC.PL0024TP.TRANS
SQRC0241	PO0124	3350	D80006	12,934	SQRC.PZ0024DA.TRANS
SQRC0242	PI0124	3350	D80006	12,934	SQRC.PZ0024DA.TRANS
SQRC0243	PI0124	3350	D80006	12,934	SQRC.PZ0024DA.TRANS
SQRC0243	PC1275CL	3350	A50005	7,293	SQRC.PC1275CL.DATA
SQRC0244	PC1274CL	3350	A81003	41,238	SQRC.PC1274CL.DATA
SQRC0244	PC1274CL	3350	A81003	108,522	SQRC.PC1274CL.INDEX
SQRC0243	PI0124	3350	D80006	12,934	SQRC.PZ0024DA.TRANS

Figure 4b. Job Step Timing Statistics

The biggest single block of time in the entire stream is clearly the DASD time in job SQRC024M. Figure 4b summarizes DASD I/O (taken from SMF Type 14, 15, and 64 records) for that job in the study runs. The largest EXCP counts are those to the data and index components of a VSAM file, SQRC.PC1274CL. According to the application programmer responsible for this jobstream, the file is usually accessed patchwise sequential, that is, each read or update usually starts with a randomly distributed record but then proceeds through a short series of the first record's successors. The tuner's recommendation was to increase the index and data buffer allocations for that file, by adding

AMP=('BUFNI=10','BUFND=5')

to the PC1274CL DD's in jobsteps SQRC0241 and SQRC0244. The recommendations were put into effect, index read activity essentially went away, and SQRC024M's execution time dropped from 252 minutes to 5.

The middle-out strategy uses modeling to estimate expected device performance and then compares observed to expected performance. Deviations from the model are investigated and diagnosed as either errors in the model or opportunities for performance improvement. This approach has been extensively discussed with respect to MVS and VM paging subsystems [15]. We shall consider another example, an application that makes heavy use of a two-cylinder checkpoint file.

The file consists of several tracks of header information (blocked 4K), followed by a large number of 4K blocks of detail data. A typical access consists of a partial header read, several random detail reads or writes, and a partial header rewrite. The file resides under the fixed heads of a 3350 volume, thereby avoiding any delays due to SEEK commands and reconnects. The rest of the volume contains data that is essentially inactive. The relevant device constants are channel speed (1.2 MB/sec), protocol time (about 2 msec per SSCH) and rotation time (16.7 msec). On the average, a target block is halfway around from the current head position so the mean rotation time ("latency") is half a rotation. For single-block data transfers we calculated

Connect time = 2 + 4K / (1.2 MB/sec)
= 5.3 msec
Disconnect time = 2 + (16.7 msec) / 2
= 10.3 msec

The RMF-reported values (24-hour average) per SSCH for this device were Connect time = 11.4 msec, Disconnect time = 24.4 msec, and Pending time = 0.7 msec. We clearly had something to investigate.

We started with the Connect time made up of error correction, control and data time. SYS1.LOGREC showed no error recovery charged against this device, so the Connect time is all control and data transfer. Command transfers, measured in bytes, are negligible relative to data measured in thousands of bytes. Apparently the application is moving [(11.4 - 2) msec * (1.2 MB / sec) / 4K] = 2.82 blocks with each SSCH. Further investigation showed that the application does indeed use chain-scheduled EXCPs against this dataset, and that 2 - 4 blocks per EXCP was not an unreasonable activity rate. Therefore, the single-block I/O assumption in the original model was in error.

We next turned to the disconnect time, which comprises SEEK time, rotational delay, and reconnect wait. SEEK time should be zero if the file is indeed under fixed heads. We double-checked the volume location and, with the IBM CE's help, made sure that the device in question was equipped with the fixed-head feature. An IEHLIST LISTVTOC report confirmed that the file occupied cylinders 1 and 2, the fixed head location. SEEK time thus was eliminated as a contributor to that 24.4 msec. Reconnect wait is a function of path busy. The 0.7 msec per SSCH Pending time applies to the downstream path (M/M/1 statistics) and is already a low number. Upstream queueing for this configuration follow M/M/1 statistics at HOS and M/M/4 statistics at the CU, so upstream path wait should be somewhat below 0.6 msec. We must still account for roughly 24 msec of rotational delay per SSCH.

The original 8.3 msec estimate was based on the assumption that the platter would have to rotate halfway around, on the average, to reach the target block. If that same latency applies to each block in a 2.8-block chain-scheduled I/O, we should see a Disconnect time of [(2.82 block) * (8.3 msec / block)] = 23.4 msec per SSCH. This is almost too good an agreement with the observed time when we add in an expected reconnect delay.

To sum up, roughly 2/3 of the response time to this file is taken up by rotational delay. That delay exists because there is no correlation between the rotational positions of two successive target blocks within any EXCP. Suppose the file uses TTR addressing. The R value for block 2 in a transmission has equal odds of being higher or lower than the R value for block 1. The process could be made more efficient by pre-sorting the blocks to be transferred. By arranging them in increasing order of R (except that duplicate R values pass to the end of the list), one increases the odds that the next delay will be of less than 1/2 rotation, and thus reduces the mean Disconnect time to less than 8.3 msec per block transferred. The tuner recommended that the application be so altered, but is still waiting for the results.

Application to Tape

Tape I/O management is in much poorer shape. Indeed, many of the relevant statistics are still ill-defined. The very meaning of some of the reported numbers varies with the tape device technology. The original definitions of the CSS variables still hold: Pending time is path wait, Connect is control and data transfer, and Disconnect clocks "device dedicated but not transferring". However, the operational definition of "device" gets a little blurry with the 3480 tape subsystem.

Let's take the simpler case first. The oldest currently supported generation of tape equipment, 3420-class devices, runs under a 3803-class control unit. The 3803 is strictly a pass-through facility. Suppose the CSS attempts to start an I/O to a 3420 drive. If a path is available straight through, connection is made and data transfer starts immediately (Connect time accumulates). If no path is available (device not READY or CU busy with another drive), then the I/O is queued in the CSS (Pending time accumulates). Disconnect time accumulates during periods when the device is executing a command that requires no data transfer (rewind, forward/backward space, etc.).

The more recent equipment, IBM's 3480 architecture [16, 17], uses a microprocessor-driven control unit with buffers that are dynamically dedicated to individual drives and I/Os. Early-ship CUs were equipped with 512K of memory, partitioned by the firmware into 64K buffers. IBM has since announced 1MB units and has even offered no-cost upgrades for the smaller memories. The buffers serve a speed-matching function. Data transfer from CSS to CU buffer takes place at channel speeds (3 MB/sec in current models). The rate from CU buffer to device depends on block size and specific function, but in general is substantially lower. The CU accepts a high-speed burst bound for drive 1, for instance, and then uses the CSS-CU path for drive 2's traffic while using a downstream path to dole out data at lower speed to drive 1.

WRITE operation starts with a channel command and data passing from the CSS to the CU's buffer(s). Pending time accumulates only if no CU buffer is available to accept the data. Connect time accumulates during the transfer to the buffer. The CU returns Channel End when the transfer to the buffer is complete. Disconnect time accumulates while the CU empties the buffer downstream. The CU returns Device End and Disconnect time stops when the transfer to the tape is complete.

READ operation is the converse. Pending time accumulates from receipt of the ORB until the I/O command can be passed to the CU. The CU frees the path (Disconnect time

accumulates) until the tape drive has loaded the CU's buffer. Then the CU reconnects (timer function switches from Disconnect to Connect) for data upload to the host.

From the CSS's viewpoint, the "device" in these operations is the CU buffer, not the target tape drive. This becomes even more apparent when one considers chain-scheduled transfers. In a WRITE, for instance, the CU may be filling buffer 2 from the CSS while simultaneously draining buffer 1 to the tape. The CU's Channel End signal is deferred until there are no vacant buffers to accept new data. CSS-measured Connect time overlaps what would otherwise have been Disconnect time.

The picture is muddied by the fact that rewind and mount wait time both register as Disconnect time for 3480, whereas they appear as Pending time for 3420. To summarize, in the context of tape devices:

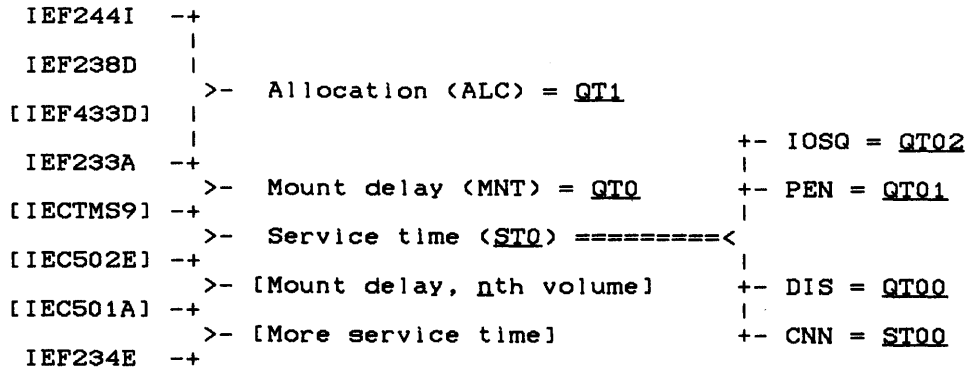
- IOSQ time - delay waiting for control of the UCB (should be none) or as a result of path sensed busy by IOS's TCH channel command.
- Pending time - delay in the CSS waiting for control of the path:
  - 3420: includes rewind and mount time.
  - 3480: CSS-to-CU path contention and buffer waits only.
- Connect time - time spent transferring control information or data:
  - 3420: to the device.
  - 3480: to the CU buffer.
- Disconnect time - time during which something is dedicated to an I/O request but not connected to the CSS:
  - 3420: "something" is the drive - time for forward or backward space by file or block.
  - 3480: "something" is the CU buffer - time waiting for buffer fill or release or allocation (which may involve rewind and/or mount waits).

The somewhat anomalous treatment of rewind and mount time points up a fundamental difference between DASD and tape. A DASD device is sharable: it can support concurrent access to one or more datasets by multiple applications. A tape device is dedicated to a single dataset and a single application: a current user must release the entire dataset before any user may access any other dataset via that device. Tape data access is thus necessarily subject to two levels of queueing. A user application must first wait for exclusive access at the dataset level. Gaining that, it must wait again for each block or burst of blocks. The two levels of queueing have timescales that may differ by a factor of a thousand or two.

Consider the effect of a design decision made early in the history of RMF. The designers worked on the assumption that the only relevant variable for tape drives is time-in-use. The device is obviously dedicated to a user during rewind or tape mount wait, so RMF tallies together all times that the device is not idle. That decision leads to collection of data that is usable for very little else than time-in-use management.

For example, suppose that allocation for batch job APRECAP has issued a message, requesting that tape volume 024978 be mounted on 3480 drive 2C4. Processing in APRECAP's address space proceeds after the mount request has appeared. Control is eventually passed to application code which attempts to OPEN the tape file. When OPEN gets control, it issues a READ against 2C4 to begin label processing. If 2C4 is not yet READY, its CMB will accumulate Pending (for 3420s) or Disconnect (for 3480s) time until the tape is mounted and made READY. Furthermore, this tape mount request is not satisfied in less than 8.3 seconds, so it is accompanied by a Disconnect time overflow. Attendant action by MIH estimates the "true" Disconnect time as some number of MIH interval expiries. RMF at current levels takes no notice of the mount wait status but accumulates all of these timings into the totals for "real" I/O. The resulting reports average together a 2-minute mount wait and a few dozen 50-millisecond READ waits and incorrectly show enormous queueing delays per I/O.

Similar logic leads to inflated tape device %busy figures from RMF/370. The numbers are so obviously improbable for block-level queueing activity that most analysts simply ignore RMF's Tape Device Activity Report, even though some of the numbers on it are valid and useful. IBM has announced that a future release of RMF will address the confounding of mount wait with I/O activity queueing times. The change will likely involve more than simply discarding active-but-not-READY times, because time-in-use is still an important capacity metric. We should see a complete recast of the report, with



Dataset response time (R1) = QT1 + ST1
= ALC + Dedicated device time
Dedicated device time (R0) = ST1 = QT0 + ST0
= MNT + Device service time
Device service time (ST0)
= Host time + SUM (IOS response time)
IOS response time (R02) = QT02 + ST02
= IOSQ + CSS response time
CSS response time (R01) = QT01 + ST01
= PEN + Subsystem response time
Subsystem response time (R00) = QT00 + ST00
= DIS + CNN

Figure 5: Tape subsystem queueing

separate columns for dataset- and block-level metrics.

If the RMF reports do not speak clearly to dataset-level delays, where can the analyst turn for data? SYSLOG is a good source for both MVS/370 and MVS/XA. Interesting data can be developed by using SAS or similar software to scan SYSLOG, looking for specific tape-related messages. The following discussion is specific to JES2 in a system with no tape-sharing software, but similar analyses can be carried out for JES3 and STAM (a shared tape drive management product of Duquesne Systems, Inc.) systems.

Each step in the mount cycle is signaled by a specific message, and each message is timestamped. All of the following messages are prefixed with the requesting job's JES job number, and all after the IEF433D message carry the device number (address) of the drive in question. The variable to be tracked is the difference in timestamps between successive messages for a given mount (see Figure 5).

Suppose all drives are in use and a new allocation request appears. Allocation issues an IEF244I message, "Unable to allocate unit...", followed by an IEF238D message, "Reply device name, wait, or cancel". There are no drives available and the job cannot be cancelled, so the operator replies "Wait". The system responds with an IEF433D message, "Wait requested -- Reply Hold or Nohold" asking whether it should release any devices the job has already received. Either way, the job sooner or later obtains a drive and issues an IEF233A message, requesting a mount of the desired volume on that device.

It would be nice to know the mount delay, the period between the IEF233A message and the moment the system senses that the correct volume is READY. Some subsystems, notably DFHSM, keep their own records of tape mount waits. IBM system software does not provide a "mount satisfied" message. However, mount time usually is very close to the time of OPEN completion. An OPEN timestamp appears in the corresponding SMF Type 14 or 15 record. SYSLOG receives an IECTMS9 message at OPEN time if (1) the tape is OPENed for output, and (2) the installation is running the UCC-1 (TMS) software (a product of UCCEL, Inc.). OPEN-for-output mount delay is a potentially useful estimate of general mount delay, but it can be biased if the system's operating procedures include special racks for scratch tapes or if Automatic Cartridge Loaders are installed on 3480s.

The dataset I/O cycle completes with a final IEF234A message, issued when the tape is demounted. The period between the IEF233A and IEF234A messages is the dataset-level service time, which is itself divisible into a series of queue and service times at the EXCP level. Figure 5 summarizes the entire sequence and makes plain the complexity of the nested queueing structures.

Before going on to consider tuning strategies applied to tape performance, we should say a few words about what can be tuned in a tape subsystem. Many of the DASD tuning options (dataset or volume reorganization, etc.) are responses to generic problems that are peculiar to DASD equipment (head movement delay, etc.). There are fewer generic problems in a tape subsystem, which is just as well because there are fewer tuning options as well.

The major options are equipment-oriented. Technology choice is one of the basics. Moving from 3420- to 3480-class equipment should affect all of the components of ST1. Reference [17] offers a particularly lucid discussion of the 3420/3480 issue with regard to performance expectations. Doubled channel speed does not necessarily yield proportionally decreased turnaround times, but one can at least expect reduced Connect times. The person making equipment decisions must recognize other concerns as well: interface requirements with other equipment, financial terms, media and storage costs, etc. Of course, the tuner is usually faced with an overriding constraint: the enterprise must work with what is on the floor.

The next tuning "parameter" is the subsystem configuration. Insufficient drives lead directly to allocation delays (QT1). Long strings of tape drives behind a small number of CUs will in general lead to greater path contention delays (QT01 and QT02). Contention problems could be addressed by adding one or more CUs to the mix and shortening the strings. Tape device waits could be addressed either by reducing service time per request or by adding more device to the existing configuration.

The 3480 CU is nearly a black box, tuning-wise. "Tuning" is the management of a scarce resource to minimize the effects of the scarcity. The scarce resource in the CU is the buffer memory. Insufficient CU buffer space leads to elevated Pending and Disconnect times (QT00 and QT01). The CU firmware has the responsibility of managing that resource. The user cannot measure CU buffer usage, but neither is there a way to influence how many buffers are to be dedicated to any drive. The only available tuning option is to ensure that the CU is equipped with the largest available memory.

Tape mount wait (QT0) is one of the most important sources of delay in the tape subsystem. Machine room geography can have a major impact on mount delay: how close are the most active tape drives to the media storage, especially for scratch tapes?

The SELTAPE= parameter in the active IEAOPTxx PARMLIB member can also have an impact on tape mount time, at least for very large tape farms. This parameter gives policy to SRM on which tape drive should be selected for the next mount. SELTAPE= RANDOM guarantees maximum exercise for the tape libarians. If device numbers are assigned with the lowest being closest to the tape console, SELTAPE=LOWEST will concentrate activity near that facility. SELTAPE=NEXT will concentrate mounts into a band that starts at low device numbers and sweeps across the farm in a regular procession.

The final option finesses the rest: reduce tape waits of all sorts by reducing tape I/O. The combination of decreasing DASD cost and increasing sophistication of data management software (IBM's DFHSM, Sterling's DMS, Innovation's ABS, etc.) has made feasible the wholesale migration of short tape datasets to DASD. Schering-Plough's Memphis Computer Center, for instance, adopted a 20-MB low threshold for tape residence. As a result of that policy, the installation was able to reduce its tape drive count by 60%, with attendant reductions in batch work delays [18].

How, then, can our tuning strategies be applied to the tape subsystem in an MXS/XA environment? The bottom-up strategy is of little use, mostly because the measurement difficulties noted above have hindered our development of useful Rules of Thumb. Some authors have proposed rules hinging on ratios of Pending or Disconnect to Connect time, but the 3480 CU's ability to overlap upstream and downstream operations limit their effectiveness. Because a tape device is dedicated to a single user (at any one time), it works perfectly well at utilizations near 100%. The 30% path busy rule is still of some use, except for a 3480 subsystem with the Dual Control Unit Communications Coupler (DCUCC) feature. The DCUCC in essence converts the 3480 subsystem from an M/M/1 to an M/M/2 server downstream and as high as M/M/8 upstream. Such a configuration can withstand path utilizations as high as 70% and still deliver as well as an M/M/1 server at 33%.

Similarly, the middle-out strategy is of limited value because of the difficulty of modeling the complex interactions of the electronic, mechanical, and human timescale queues. The M/M/c relations do offer a way to estimate drive count requirements, starting with an estimated mean dedicated device time (ST1) and a desired mean allocation wait (QT1). Figure 6 shows a family of QT1/ST1 versus AT1/ST1 curves (calculated with the code in Appendix 1) for different values of c (tape drive count), where AT1 is the mean mount request arrival rate. Dividing both interarrival time and queue time by service time allows the same set of curves to be used for any

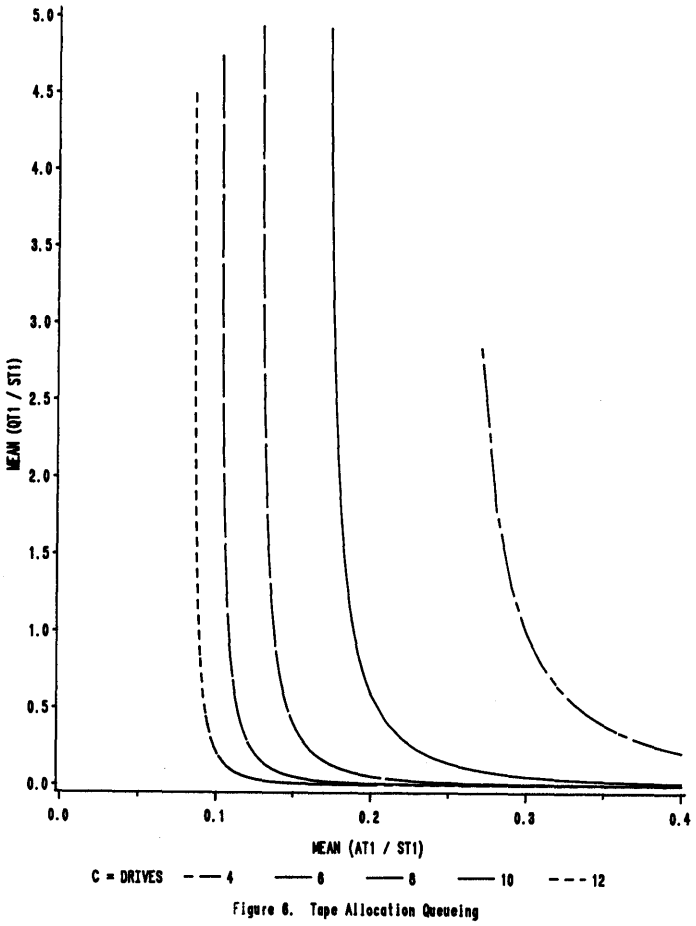


Figure 6. Tape Allocation Queueing

average service time. For instance, the curves say that if an installation's mean ST1 is 5 minutes and its mean AT1 is 1 minute (AT1/ST1 = 0.2), 4 drives will be unable to handle the load, 6 would give a mean (but risky) QT1 near 2.5 minutes (0.5 * ST1), but 8 or more drives should provide acceptable service.

These predictions should be taken with a good deal of salt. The major problem is that the M/M/c model assumes that arrivals and service times are each Markovian, that is, no arrival or service time depends upon any other arrival or service time. This is not the case for real batch workloads, which usually mount two or even three tapes simultaneously for the length of the same job step. The correlated activity of n servers effectively reduces the server count by (n - 1).

The top-down strategy is the most useful here, because it concentrates on locating problems as perceived by the user. During the SQRC subsystem performance analysis of Figure 4, for example, the analyst noticed that one job step had an unusually large disparity between the total execution time and the total of allocation, ENQ, CPU (expanded by the mean Multi-Programming Level), DASD and tape times. Further probing showed that one tape DD's Connect time per EXCP was only a tenth that of the installation mean for tape datasets.

Short Connect time indicates a small block size, and sure enough, this dataset had 800-byte blocks instead of the installation-standard 20K blocksize. Some relevant numbers are collected in Figure 7. The small block size means that only a small number or records are transmitted with each EXCP. Since each EXCP carries an overhead penalty (the sum of IOSQ, Pending, and Disconnect time, averaging to 20 msec per EXCP system-wide exclusive of mount and rewind time), small block size also means high overhead. The last entry in the "Prior" column shows that only 6% of the I/O time against this dataset was actually spent transferring data.

	Expression	Prior	Future
File size, MB	F	26	26
Block size, KB	B	0.8	20
Channel speed, MB/sec	S	3	3
KBytes per 5-block EXCP	5B	4	102.4
EXCPs to read file	E = 1000 * F / 5B	6500	254
Connect msec per EXCP	C = (5B / S) + 2	3.3	32
Total Connect sec	C * E / 1000	21.6	8.1
I/O msec per EXCP	T = 20 + C	23.3	52
Total I/O time	T * E / 1000	151	13.2
I/O efficiency	100 * (5B / S) / T	5.6	57.7

Figure 7. Improving I/O by Reblocking

Suppose the dataset were reblocked from 10 to 256 records per block (blocksize 20480). The expected I/O timings are shown in the "Future" column of Figure 7. The reblocking would raise the I/O efficiency to nearly 60% and we would expect to reduce the job step's run time by over hundred seconds. (The actual magnitude of the reduction would depend on how much of the job's I/O to this dataset overlaps other processing.) The file was reblocked on the analyst's recommendation, and the job step's execution time in fact dropped by 170 sec. Expediting the tape I/O so sped up processing that the job suffered far less paging and swapping than it had with the smaller blocksize.

Summary

The advent of XA moved MVS's I/O measurement philosophy from resource measurement to time measurement. That reorientation has facilitated the performance analyst's ability to apply the top-down tuning strategy, the most efficient way to locate and relieve performance bottlenecks. Relative levels of measurement sophistication make systematic tuning most applicable to DASD subsystems, less applicable to tape and network problems.

References

[1] - Allen, A. O., Probability, Statistics, and Queueing Theory, Academic Press, New York NY (1978).

[2] - Anonymous, MVS/370 Resource Measurement Facility (RMF) Reference and User's Guide (Document SC28-0740), IBM Corporation, Poughkeepsie NY (1980).

[3] - Salla, A., RMF - Field Description and Analysis. Version 1. MVS/SP 1.3.0 (Document GG24-1564), IBM World Trade Systems Center, Poughkeepsie NY (1982).

[4] - Wicks, R. J., MVS/XA: An Overview (Document GG22-9303), IBM Washington Systems Center, Gaithersberg MD (1982).

[5] - Anonymous, IBM System 370/Extended Architecture Principles of Operation (Document SA22-7085), IBM Corporation, Poughkeepsie NY (1985).

[6] - Anonymous, MVS/Extended Architecture Resource Measurement Facility (RMF) Reference and User's Guide (Document LC28-1138), IBM Corporation, Poughkeepsie NY (1983), Chapter 11.

[7] - Anonymous, MVS/Extended Architecture System Programming Library: System Management Facilities (SMF) (Document GC28-1153), IBM Corporation, Poughkeepsie NY (1986).

[8] - Woods, J., MVS/Extended Architecture Resource Measurement Facility (RMF) Monitor III (Document GG22-9350), IBM Washington Systems Center, Gaithersburg MD (1984).

[9] - Anonymous, MVS/Extended Architecture System Programming Library: Initialization and Tuning Guide (Document GN28-1056), IBM Corporation, Poughkeepsie NY (1986).

[10] - Beretvas, T., "DASD Performance Analysis Using Modeling", CMG '85 Proceedings, 749-760 (1985).

[11] - Olcott, R., "Management of Online Storage in a Multi-Processor Environment", CMG '85 Proceedings, 791-812 (1985).

[12] - Reed, M. L., "Managing DASD Performance to Satisfy Workload Requirements", CMG XII Proceedings, 119-126 (1981).

[13] - Mullen, J. W., "MVS Performance and Resource Measurement Facility (RMF) Data Analysis", CMG '86 Proceedings, 803-812 (1986).

[14] - Juzwik, F., "Tuning Methodology Based on Performance Management", CME Compendium VIII, SHARE, Inc., Chicago IL, 43-52 (1986).

[15] - Beretvas, T., "How to Analyze the Effectiveness of Your Paging Subsystem", CME Compendium VIII, SHARE, Inc., Chicago IL, 53-92 (1986).

[16] - Anonymous, 3480 Magnetic Tape Subsystem Planning and Migration Guide, Document GC35-0098), IBM Washington Systems Center, Gaithersburg MD (1986).

[17] - Middleton, G. F., and Taylor, L. B., IBM 3480 Tape Subsystem: Performance Considerations, Document GC22-9335), IBM Washington Systems Center, Gaithersburg MD (1985).

[18] - Olcott, R., Total Storage Management at Schering-Plough, SHARE Proceedings, SHARE, Inc., Chicago IL, in preparation.

Appendix 1. SAS Code for M/M/c Curves

```
PROC FORMAT;
  * calculate factorials by table look-up;
  VALUE fact  1='1'  2='2'  3='6'  4='24'  5='120'  6='720'
              7='5040'  8='40320'  9='362880'  10='3628800'
              11='39916800'  12='479001600';

DATA points (KEEP = iat c que);
* SAS data set will contain que as a function of c and iat;
  ARRAY rcs (*) rc1 - rc12;
  ARRAY sss (*) ss1 - ss12;
  ARRAY ccu (*) ccu1 - ccu12;
  RETAIN  one 1  stm 1;
  LABEL stm = 'Service time';
  LABEL iat = 'Inter-arrival time';
  LABEL que = 'Mean queue time';
  LABEL c   = 'Server count';
  DO  iat = .0842, .0844, .0846, .0848,
            .001 TO .2 BY .001,  .2 TO 2 BY .02;
    * points chosen for smooth curves through que = 5;
    LINK formula;
    END;
  RETURN;

formula:
* transform of expression in [1], Appendix C, Table 5, p357;
  lambda = one / iat;                          /* Arrival rate */
  mu = one / stm;          /* Mean service rate per server */
  u = lambda / mu;                    /* Traffic intensity */
  ss = one;
  DO  jj = one TO 12;
    fact = INPUT (PUT (jj, fact.),9.);     /* jj factorial */
    rcs(jj) = u**jj / fact;                /* typical term */
    ss + rcs(jj);
    sss(jj) = ss;
    END;
  c = one;                              /* Single server case */
  IF  u < one
  THEN  que = MIN (9999, stm * u / (one - u));
  ELSE  que = 9999;
  OUTPUT;
  DO  c = 2 TO 12 BY 2;                /* multi-server cases */
    rho = u / c;                  /* utilization per server */
    IF  rho < one  THEN  DO;
      ccu(c) = rcs(c) / (rcs(c) + (one - rho) * sss(c - 1));
      que = MIN (9999, ccu(c) * stm / (c * (one - rho)));
      END;
    ELSE  que = 9999;
    OUTPUT;
    END;
  RETURN;
```

FROM SIMPLE REGRESSION TO ANALYSIS OF VARIANCE AND BEYOND--PAINLESSLY

Ann C.Pool
Planning Research Corporation

ABSTRACT

An understanding of the theory of simple linear regression and confidence intervals can be expanded to gain invaluable additional information. In a few steps an analyst can go from a conceptual understanding of linear regression to an understanding of analysis of variance and analysis of covariance. This paper attempts to demystify linear regression models and addresses some of the its applications and common misapplications. The general linear regression model and its assumptions, confidence intervals and prediction intervals, and the formulation of linear regression models that can be used for prediction, classification of data, and modeling growth are discussed.

1. INTRODUCTION

Often one hears "I tried every test in the book" or " I don't use statistics, because I don't trust them ". What both of these people are saying is that they do not understand statistics. Generally in statistics only one or two tests or series of tests are appropriate in a given situation. Statistics can be a powerful tool if the underlying assumptions and restrictions are understood. Perhaps one of the greatest weaknesses of many computer and business professionals is that they do not understand statistical methods and procedures. The purpose of this paper is to demystify linear regression models and to address some of its applications and common misapplications. This paper does not set out to suggest computational methods or the abandonment of standard computational procedures.

Linear regression can be used as a statistical model of a given situation. Linear regression offers a rich and full language for interpreting relationships into mathematical equations and statistical procedures. Regression analysis allows formulation of models that can be used to explore the functional relationships, and to predict future conditions. Just like any other modeling process, using linear regression requires definition of the problem, identifying appropriate input and output variables, finding appropriate mathematical representation of the relationships, and testing the model.

2. STRAIGHT-LINE REGRESSION

Linear regression divides all variables into two types: independent variables and dependent variables. Independent variables are the predictor variables. Independent variables can be set or observed but not controlled. They drive the equation. Dependent variables are the response variables. These variables represent the consequence of independent variables. They are the result of the equation. For example, in the equation:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 \qquad (1)$$

Y is the dependent variable and X is the independent variable. Equation (1) is a second order (in X) linear (in the β's) equation.

Equations which are not linear can be transformed into a linear equations or approximated by one.

Consider, the following equation:

$$Y = a\, X^b \qquad (2)$$

Y and X represent known values, a and b are unknown values. This equations is not linear, because b is an unknown. However, it can be expressed in linear form as

$$\log(y) = \log(a) + b \log(X) \qquad (3)$$

Equation (3) is a first order (in log(X)) linear equation. log(a) and b constant parameters that can be calculated using linear regression.

2.1 ASSUMPTIONS OF LINEAR MODELS

Like any statistical test or model there are certain assumptions associated with linear regression. If X represents the independent variable(s) and Y is the dependent variable, the assumptions can be stated as follows:

1. For any fixed value of the variable X, Y is a random variable with a certain probability distribution.
2. The Y values are statistically independent of one another.
3. The mean value of Y is a straight line function of X.
4. The variance of Y is the same for any X.
5. For any fixed value of X, Y has a normal distribution.

Each of these assumption has an important implication for linear regression models. Consider each of the assumptions individually.

1. For any fixed value of the variable X, Y is a random variable with a certain probability distribution. The mean of this populations is denote by $m_{X|Y}$ and $s^2_{X|Y}$ is the variance of the population. "Y|X" means the random variable Y depends on the value of X.
2. The Y values are statistically independent of one another. This means that information about one value of Y gives no information about other values of Y. For example, consider the amount of disk space used on a system measured at various points in time. If there is no procedure

in operation to remove old files from the system between the two measurements, then the amount of disk space would not be statistically independent. This assumption is often violated. Special techniques for handling these situations include time series analysis and growth curve analysis.

3. The mean value of Y is a straight line function of X. This assumption can be written as

$$m_{Y|X} = \beta_0 + \beta_1 X \qquad (4)$$

where β_0 and β_1 are the intercept and slope of the regression line respective. An alternate form of this equation is

$$Y = \beta_0 + \beta_1 X + E \quad \text{or} \quad E = Y - (\beta_0 + \beta_1 X) \qquad (5)$$

where E is a random variable which has a mean 0 at a fixed X. The variable E describes the distance from an individual values of Y to the regression line. E represents the residual of the linear regression.

4. The variance of Y is the same for any X. This is the assumption of homoscedasticity or the "same scattering". This assumption is written as

$$s^2_{Y|X} = s^2 \qquad (6)$$

Since $\beta_0 + \beta_1 X$ is a constant for a given value of X, s^2 is also the variance of E.

5. For any fixed value of X, Y has a normal distribution. This assumption makes it possible to evaluate the statistical significance of the relationship between X and Y. It is important to note that there is a certain amount of leeway in this assumption. This is called the "robustness" of the regression analysis. This assumption also implies that E has a normal distribution with a mean of 0 and a variance of s^2. If this assumption can not be met by the original data the values of Y can be "transformed". Common transformations include using logarithmic functions, reciprocals, and square root functions.

2.2 CONFIDENCE INTERVALS AND PREDICTION INTERVALS

Confidence intervals and prediction intervals are very useful tools. They allow the analyst to evaluate the value of a parameter or calculated value and the quality of the estimate. The center of a confidence interval or prediction interval is generally the calculated value of the parameter. The width of the interval represents the quality of the estimate. The wider the interval for a given level of confidence, the less certain the estimate. Confidence intervals consists of boundaries between which there is a particular level of confidence that the unknown value will actually lie. For example, a 95% confidence interval for a fixed, unknown value w consists of an upper limit and a lower limit so that, there is a probability of 0.95 that the true value of w is contained in the interval. A prediction interval is used to predict a value of the function under consideration that was not part of the original computation.

In considering linear regressions, confidence intervals are associated with the calculated parameters (the β's) and the calculated estimates of Y. Table 1 shows confidence intervals for a simple regression of the form

$$Y = \beta_0 + \beta_1 X \qquad (7)$$

Notice that in Table 1 there is a confidence interval for $m_{X|Y}$ and one for Y_{x_0}. The confidence interval for $m_{Y|X}$ is for use with values of X that were used in the calculation of the regression. Y_{x_0} is used with values of X that were not used in calculation of the regression. This confidence interval is sometimes called a prediction interval. Notice that these equations are very similar. The confidence interval for $m_{Y|X}$ is

$$Y = m_{Y|X} \pm t_{(n-2,\ 1-a/2)}\ S_{X|Y} \sqrt{\frac{1}{n} + \frac{(X_0-\bar{X})^2}{(n-1)S_X^2}} \qquad (8)$$

The confidence interval for Y_{x0} is

$$Y = Y_{x0} \pm t_{(n-2,\ 1-a/2)}\ S_{X|Y} \sqrt{\frac{1}{q} + \frac{1}{n} + \frac{(X_0-\bar{X})^2}{(n-1)S_X^2}} \qquad (9)$$

where q is the number of Y values expected at X_0. That is, if at X_0 there is to be one value of Y, then the confidence interval of Y_{x0} is much wider than the confidence interval of $m_{Y|X}$. As the number of Y values expected the X_0 increases, the confidence interval of Y_{x0} approaches the confidence interval of $m_{Y|X}$. Figure 1 shows the plots of a regression line, and the associated confidence intervals for $m_{Y|X}$ and for Y_{x0}.

Often it is more important to know the values of the independent variable associated with a confidence or prediction interval. For example, if time is the independent variable (X) and cost is the dependent variable (Y). It may be useful to know the earliest and latest points in time when a certain cost might be expected. The interval between these values is a fiducial interval. Fiducial intervals for a confidence interval and a prediction interval are shown in Figure 1. The fiducial interval associated

with the confidence interval is given by

$$X = \bar{X} + \frac{b_1 (Y_0 - \bar{Y})}{b_1^2 - \frac{t^2 S_{x|y}^2}{S_{xx}}} \pm \frac{t_{(n-2,\ 1-\alpha)} S_{y|x} \sqrt{\frac{(Y_0 - \bar{Y})^2}{S_{xx}} + \frac{b_1^2}{n} - \frac{t^2 S_{x|y}^2}{n S_{xx}}}}{b_1^2 - \frac{t^2 S_{x|y}^2}{S_{xx}}} \quad (10)$$

The fiducial interval associated with the prediction interval is

$$X = \bar{X} + \frac{b_1 (Y_0 - \bar{Y})}{b_1^2 - \frac{t^2 S_{x|y}^2}{S_{xx}}} \pm \frac{t_{(n-2,\ 1-a)} S_{y|x} \sqrt{\frac{(Y_0 - \bar{Y})^2}{S_{xx}} + \left(\frac{1}{n} + \frac{1}{q}\right)\left(b_1^2 - \frac{t^2 S_{x|y}^2}{S_{xx}}\right)}}{b_1^2 - \frac{t^2 S_{x|y}^2}{S_{xx}}} \quad (11)$$

These intervals do not exist for every linear regression at every value of the dependent variable. To be of practical value β_1 must be significantly different from 0, that is 0 is not in its confidence interval, and $(t^2 S_{x|y}^2)/S_{xx}$ should be smaller then about 0.20.

In looking at the regression parameters, it is important to note that since the estimate of the β's are not independent, one cannot combine directly the confidence intervals of b_0 and b_1. The joint confidence interval for b_0 and b_1 is the confidence interval for the combinations of the β's. Using matrix notation, the joint confidence interval has the form

$$(\beta - b)' X'X (\beta - b) \leq p s^2 F(p, v, 1-a) \quad (12)$$

This interval results in a elliptical confidence region for two parameter equations. Figure 2 shows the 95% confidence interval for b_0 and b_1 and the joint confidence region for b_0 and b_1. The point A is in the confidence intervals for both b_0 and b_1 . Point A is clearly satisfactory when considering only b_0 or only b_1. However, when considering both b_0 and b_1, point A is not satisfactory.

3. LINEAR REGRESSION MODELS

Any modeling, including statistical modeling, is an ongoing process beginning with a question or an idea, simplification of the idea to a mathematical form, evaluation of the mathematical form, and analysis of the results. In the case of statistical models, each model is associated with at least one statistical test which may disprove a hypothesis. Linear regression models can be used to model, among other things, the differences among groups and the relationships among variables.

In developing a linear regression model the first steps are generally to graph the data, examine the graph and hypothesis one of the primary relationships among the variables examined, and then test the hypothesis. A simple regression model is first constructed and tested. Other factors are then added to reduce the residual error (E) until either the remaining factors do not significantly reduce the error or all hypothesises have been tested. This process will require several iterations. For example, consider a computer system with four disk drives that are run during 3 shifts. The users are complaining about slow disk access. One of the questions that must be answered is "Is the disk access time related to the drive on which the file being accessed is found or to the shift when the file is accessed?" At least three tests are required: one relating access time to the drive, one relating access time the shift, and one relating access time to both shift and drive.

3.1 MODELING RELATIONSHIPS AMONG GROUPS

Linear regression models can be used to examine the difference among some continuous variable such as a measurement made on groups of individuals. The model can include information on nominal factors, such as membership in groups, and continuous factors. Nominal factors contain categories or groups called levels. Factors may be either fixed or random. A fixed factor is one in which has all of the relevant levels of interest. A random factor is one which has only a sample of the levels available from some larger classification. For example, suppose a manufacturer conducts a test of a sample series of disk drives selected at random. If the measurements are grouped by the drive, and the information is applied to all disk drives manufactured, the disk drives represent a random factor. If the same test is run on all drives at a site and the results are only applied at that site, the disk drives represent a fixed factor.

The problem is now how to represent the relationships as in a linear equation. Since there is no restriction on the selection of the independent variables (X's) in a linear model, they can be used to represent membership in a group. Since the outcome a linear model is the mean of the dependent variable (Y) under the conditions in the model, i.e.,

$$m_{Y|X} = \beta_0 + \beta_1 X_1 + \ldots + \beta_{k-1} X_{k-1} \quad (13)$$

it is possible to choose the values of X so that the β's represent the means of the groups. That is, the mean response of any group is a linear combination of the β's. The responses of the groups 1 to k-1 are explicitly represented. The kth group is implicitly represented. If the kth group had been explicitly represented, the

computation of the linear regression would have been impossible.

The regression model can be re-written as

$$Y_{i,j} = \mu + \sum_{i=1}^{k-1} \beta_i X_i + E \quad (14)$$

where $Y_{i,j}$ is the dependent or response variable. It is often a measurement made under known or controlled situations. $Y_{i,j}$ is the "jth" element of group i. μ is the true value of the mean of some fixed group. β_i represents the difference between μ and the mean of the $Y_{i,j}$'s in group i. X_i represents membership in group i. E is the random error or the difference among individuals. There are a number of ways of coding membership in a group. In one scheme uses the following:

$$X_i = \begin{cases} 1 \text{ if group } i \\ 0 \text{ otherwise} \end{cases}, \quad i=1, 2, \ldots k-1 \quad (15)$$

In this coding scheme, the means of the groups are compared to a base group. For example, if there are 4 groups, the equation (14) can be interpreted as shown in table 1. In this example the base group is group 4. In an alternative coding scheme

$$X_i = \begin{cases} 1 \text{ if group } i \\ -1 \text{ if group } k \\ 0 \text{ otherwise} \end{cases}, \quad i=1, 2, \ldots k-1 \quad (16)$$

This coding scheme compares the means of the individual groups to the over all mean or the grand mean. An example of this coding scheme for four groups is shown in Table 2.

The are two different tests associated with this modeling depending on the type of factor being investigated. If the factor under consideration is fixed, the null hypothesis is H_0: $\beta_1 = \beta_2 \ldots = \beta_k$ versus H_1: not so. The English translation of this hypotheses is: the assumption to be disproved is there is no differences among the means of the groups versus, at least one means of the groups is different from the others. The results of an analysis of variance are generally analyzed through an ANOVA table such as the one shown in Figure 3. The table shows the error associated with different portion of the model. The table shows the degrees of freedom (df), the sum of the squared errors (SS), and the mean squared error (MSE). The degrees of freedom are the number of independent pieces of information involving k independent numbers. The sum of the squared error shown between groups

$$SS_b = \sum_{i=1}^{k} J_i \ (\hat{Y}_i - \bar{Y})^2 \quad (17)$$

represents the error that is introduced by estimating the mean of the measurements for each level in the factor by the mean of all of the measurements in all of the levels.

The mean squared error for the groups or variance

$$s_b^2 = \sum_{i=1}^{k} J_i \ (\bar{Y}_i - \bar{Y})^2/(k-1) \quad (18)$$

is the sum of the squared error divided by the degrees of freedom with k-1. The error within groups

$$SS_w = \sum_{i=1}^{I} \sum_{j=1}^{J_i} (Y_{ij} - \hat{Y}_i) \quad (19)$$

in found by comparing the actual measurement to the measurement estimated by the model $\hat{Y}$ and the actual measurement.

The mean squared error within groups

$$s_w^2 = \sum_{i=1}^{I} \sum_{j=1}^{J_i} (Y_{ij} - \hat{Y}_i)/(n-k) \quad (20)$$

is the variance associated with the model. The question becomes which is larger, the average error associated with using the overall mean or the average error associated with using the model.

These two numbers are compared by forming a ratio (a test-statistics)

$$F = \frac{s_b^2 \Leftarrow \text{mean error for the model using only the grand mean to represent the mean of each group}}{s_w^2 \Leftarrow \text{mean error for the model using mean of each group to represent each measurement}} \quad (21)$$

If there is more error associated with the model including the factors then with a model including only the means then the ratio will be greater than one. In order to determine how much greater than one is significant, the ratio is compared to be statistical distribution called the F-distribution. This distribution takes into account the number of parameters being estimated and the number of measurements used in the estimation. If the factor is random, it is assumed that the β's are distributed in a normal distribution with a mean of 0 and variance of $\sigma\beta$. The null hypothesis $\sigma\beta = 0$. This hypothesis implies that there is no difference among the groups. To make this test, the same ratio is compared to the the ratio $F = s_b^2 / s_w^2$ (test-statistic) to the suitable percentage point of the F(k-1, n-k) distribution (F-statistic).

Both of these schemes will give exactly the same test statistic (F-statistic) . The difference lies in the interpretation of μ. In the first scheme μ is the mean of a base group that is used to compare with the means of the other groups. In the second scheme μ is the grand mean and all group means are compared to it.

If the first scheme is used, additional information about the relationship to some base group is gained. If the second scheme is used, then information relative to the grand mean is gained. Additional information can be gained from either scheme by examining the confidence intervals of the calculated parameters (β_i's). Both schemes are suitable for fixed factors.

There are two methods of reporting the results of a statistical test. Traditionally, a predetermined percentage point is chosen from the appropriate distribution and a comparison is made. In the alternate method of reporting, the results of the test are reported as the percentage point of the distribution associated with the ratio. This allows the reader to access the risk that he is willing to take in accepting the hypothesis. This second approach is not recommended for use with a audience who is not familiar with statistics or who is not willing to accept the responsibility for the interpretation. However, many managers are willing to accept more than the traditional 5% error on a test and this allows them the opportunity to make the decision to reject or accept the hypothesis.

This model in equation 14 is called a One-way Analysis of Variance. The groups which are different can be determined by examining the confidence intervals associated with the parameters that contribute to the mean of the group. If the confidence interval contains zero (0) then the group mean is not statistically significantly different from μ. However this method has some draw backs. First making a number of tests and combining the results to reach a conclusion, reduces the significance(meaningfulness) of the group of tests. Secondly, the parameters are not strictly independent. It becomes difficult to make multiple comparison confidence intervals. Alternatively, post priori tests can be performed to determine which group is different. These tests include LSD (least significant difference) approach, Tukey's method, Scheffé's method, and orthogonal contrasts. Each of these tests combines into a single hypothesis the relationships among the means of the groups. These tests reduce the amount of overall error used in evaluating the relationship among the groups.

3.2 MODELING RELATIONSHIPS AMONG TWO GROUPINGS OR FACTORS

Often one grouping will only partially explain the relationships among the subjects under consideration. If two groupings of the subjects are used then the groupings may be called row and column factors or blocks and treatments. In either case each individual is defined by one and only row factor and one and only one column factor or by one block and one treatment. The intersection of a row factor and a column factor is call a cell. The individuals in a cell have some measurable characteristic that is partially due to the row factor and partially due to the column factor. The mean value of the measurements in a cell has a component or part that is due to the row factor, a part that is due to the column factor, and a part due to the interaction between the two factors. For example, in evaluation disk access time, not only the drive may be important, but the shift during which the measurements are made may be important. The relationship between the shift and the drive may vary from drive to drive and shift to shift. This relationship may be modeled as two factors and an interaction. The factors are the disk drive and the shift. This model is called a two-way analysis of variance. This relationship is shown schematically in Figure 4.

The basic model for this situation is

$$Y = \mu + \sum_{i=1}^{k-1} \alpha_i X_i + \sum_{j=1}^{b-1} \beta_j Z_j + \sum_{i=1}^{k-1} \sum_{j=1}^{b-1} \gamma_{ij} X_i Z_j + E \qquad (22)$$

where X_i represents the membership in the group with the row factor of level i and Z_j represents membership in the group with the column factor. α_i and β_j are the components of the mean of Y due to the factors. γ_{ij} is the component of the mean due to the interaction between the two factors. One scheme used to represent membership is [1]

$$X_i = \begin{cases} 1 \text{ if level i of row factor} \\ -1 \text{ if level k of row factor} \\ 0 \text{ otherwise} \end{cases} \qquad (23)$$

$$i = 1, 2, \ldots k-1$$

and

$$Z_j = \begin{cases} 1 \text{ if level j of column} \\ -1 \text{ if level b of column factor} \\ 0 \text{ otherwise} \end{cases}$$

$$j = 1, 2, \ldots b-1$$

This scheme compares each cell to the grand mean. This is illustrated in Figure 5.

There are three null hypothesises to be tested in this model:

1. H_o: The row factor has no effect.
2. H_o: The column factor has no effect.
3. H_o: The interaction has no effect.

The testing process is further complicated by the fact that a factor may by either fixed or

[1] The first scheme shown with the One-way ANOVA may be used if the interaction term is not included in the model.

random. This leads to the following possible scenarios:

Both factors are fixed

Both factors are random

The row factor is fixed and the column factor is random

The row factor is random and the column factor is fixed.

Thus there are four series of tests that might be performed. The analyst will choose the series of tests based on the situation. For example, in considering the disk access time problem for a given site, the drives are a fixed factor, and the shift may be either a fixed factor or a random factor. If the information from the test is to be applied to historic situations only, then the factor may be fixed. However, if the information is to be applied to all situations, past and future, the factor is random. There are different test-statistic used depending on the type of factors involved. These tests are summarize in Figure 6. Notice that there is a difference in the test statistics for fixed and random effects. Also note that the sums of squares are calculated for cells with the same number of members in each one.

3.3 ANALYSIS OF COVARIANCE

At times discrete factors are not enough to explain the relationships among variables. Continuous variables can be added to an ANOVA to account for such relationships. This model is call an Analysis of Covariance. The linear regression representation of this model is

$$Y = \mu + \sum_{i=1}^{p} \alpha_i X_i + \sum_{j=1}^{b-1} \beta_j Z_j \qquad (24)$$

where X_i is a continuous variable and Z_j represents membership in a group. Since this model represents only a one-way analysis, either scheme for representing membership could be used. If a two-way analysis with interaction is used only the second scheme could be used. It is important to note that this test is used under two conditions. First, if groups appear to significantly different under conditions where it is hypothesized that they should not. Alternately, it may be used if groups do not appear to be significantly different but should. For example, if the access time appears to be significantly different when grouped by shift, adding the covariant of disk location to the model may make the grouping not significant.

Analysis of this model is performed in two steps. First the relationship between the dependent variable (the Y) and the covariants (the X's) is confirmed by testing the statistical significance of the model

$$Y = \mu + \sum_{i=1}^{p} \alpha_i X_i \qquad (25)$$

This is confirmed by comparing the model in (25) with p=1 to the model with p=0 using a F-statistic similar to the process discussed before. Then compare the model with p=2 to the model with p=1. This process is continued until the test results do not lead to the rejection of the null hypothesis or all of the hypothesized convariants have been added to the model. The second step is to compare the modeling including the factors (25) to the model not including the factors (26). If the null hypothesis is rejected then the groups are said to be significantly different when "controlling for covariants".

3.4 THE COMPLICATIONS OF ANALYSIS OF VARIANCE

There are several issues in using analysis of variance models. The first is the number of subjects or measurements in a cell. The examples shown above assume an equal number of subjects per cell. There are computation methods used for models with unequal cell size. These methods are discussed in the references. Many statistical computer packages provide analysis of variance routines which may or may not deal with unequal cell size. Sometimes an analyst can use the regression routines to preform analysis of variance. Another issue is the number of factors and covariants to be used. The number of variables used to explain dependent variable should be held to be minimum.

3.5 GROWTH MODELS

Growth models deal with change over time. They can be approached in the same manner as the analysis of variance models--observe, hypothesize, test. There are two important questions to be answered:

Can the relationship be modeled under the restrictions of linear regression?

Can the shape growth curve be modeled using linear regression?

Often the second principle of regression (The Y values are statistically independent of one another.) is often violated in using a model, for example

$$Y_i = \mu + \beta t_i \qquad (26)$$

If Y_2 can be predicted by knowing Y_1 then some change must be made in the model. This condition can be eliminated by using $Y_2 - Y_1$ as the dependent variable. Then the model is

$$(Y_2 - Y_1) = \mu + \beta t_i \qquad (27)$$

Under many conditions, growth is not a straight line. Under these condition a transformation of the data may be used. For example, if the data can be model as

$$Y = bt^a \qquad (28)$$

This model can be transformed using the logarithmic function to

$$\log(Y) = a \log(t) + b \qquad (29)$$

Either of these models may be expanded to include other independent variables.

4. SUMMARY

It is possible to model many situation using linear regression models including the relationships among groups, growth, and logarithmic relationships. Independent variables may be either discrete or continuous. Linear regression models can be used for the applications discussed in this paper as well as such diverse applications as discrete multivariate analysis and time series analysis. The modeling process using a statistical approach is the same as using any other modeling process. All linear regression can be reduced to the comparison of a model to a simpler one. Knowledge of the methods lead to predictions, understanding of relationships, and the ability to evaluate statements and claims.

References

Draper,Norman and Harry Smith, Applied Regression Analysis, edition 2,1981.

Kleinbaum, David and Lawrence L Kupper, Applied Regression Analysis and Other Multivariable Methods, 1978.

Table 1
Confidence Intervals and Prediction Intervals for
$Y = b_0 + b_1 X$

Parameter	100(1-a)% Confidence Interval
β_1	$\beta_1 \pm t_{n-2,1-a/2} \frac{Sx\|y}{S_x^2\sqrt{n-1}}$
β_0	$\beta_0 \pm t_{n-2,1-a/2}\, Sx\|y \sqrt{\frac{1}{n} + \frac{\bar{X}2}{(n-1)S_x^2}}$
$m_{y\|x_0}$	$\hat{Y} \pm t_{n-2,1-a/2} S_{Y\|X} \sqrt{\frac{1}{n} + \frac{(X_0 - \bar{X})2}{(n-1)S_x^2}}$
Yxo, Prediction Interval for q observations , not a parameter	$\hat{Y} \pm t_{n-2,1-a/2} S_{Y\|X} \sqrt{\frac{1}{q} + \frac{1}{n} + \frac{(X_0 - \bar{X})2}{(n-1)S_x^2}}$

where

n is the number of points used in the calculation
q is the number of points to be estimated
$t_{n-2,1-a/2}$ is 100(1-a)% point of the t distribution with n-2 degrees of freedom

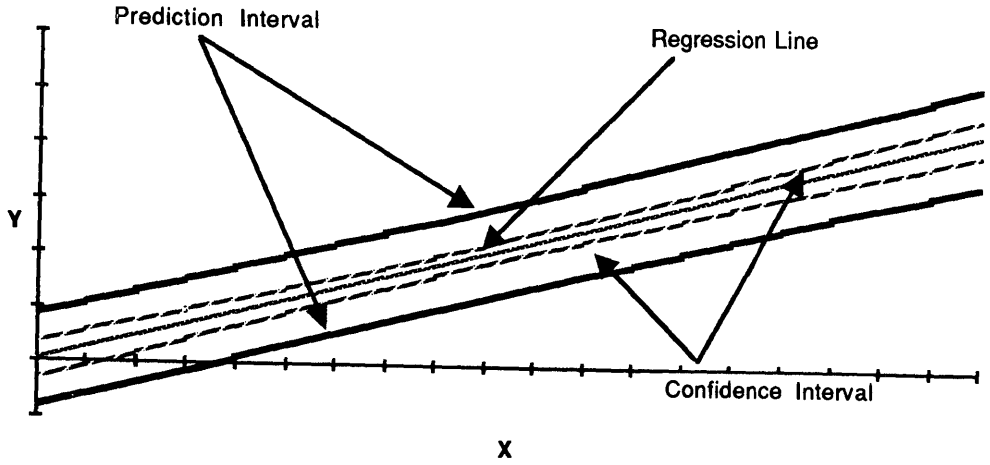
$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

$$SS_x = \sum_{i}^{n} (X_i - \bar{X})^2 \qquad S_x^2 = \frac{1}{n-1} SS_x$$

$$S_y^2 = \frac{1}{n-1} \sum_{i}^{n} (Y_i - \bar{Y})^2 \qquad S_{Y|X}^2 = \frac{(n-1)}{(n-2)} (S_Y^2 - \hat{b}_1^2 S_x^2)$$

Figure 1

95% Prediction and Confidence Intervals for a Simple Linear Regression



95 % Fiducial Intervals

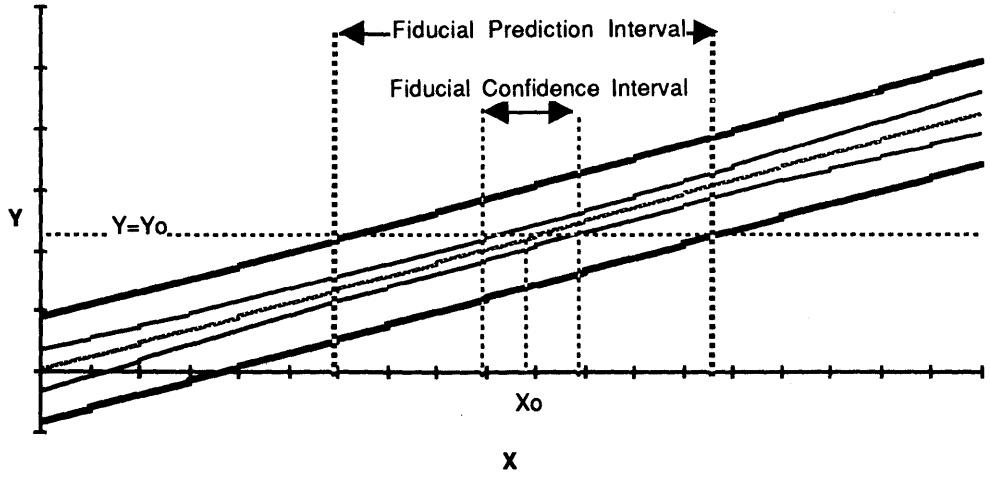


Figure 2

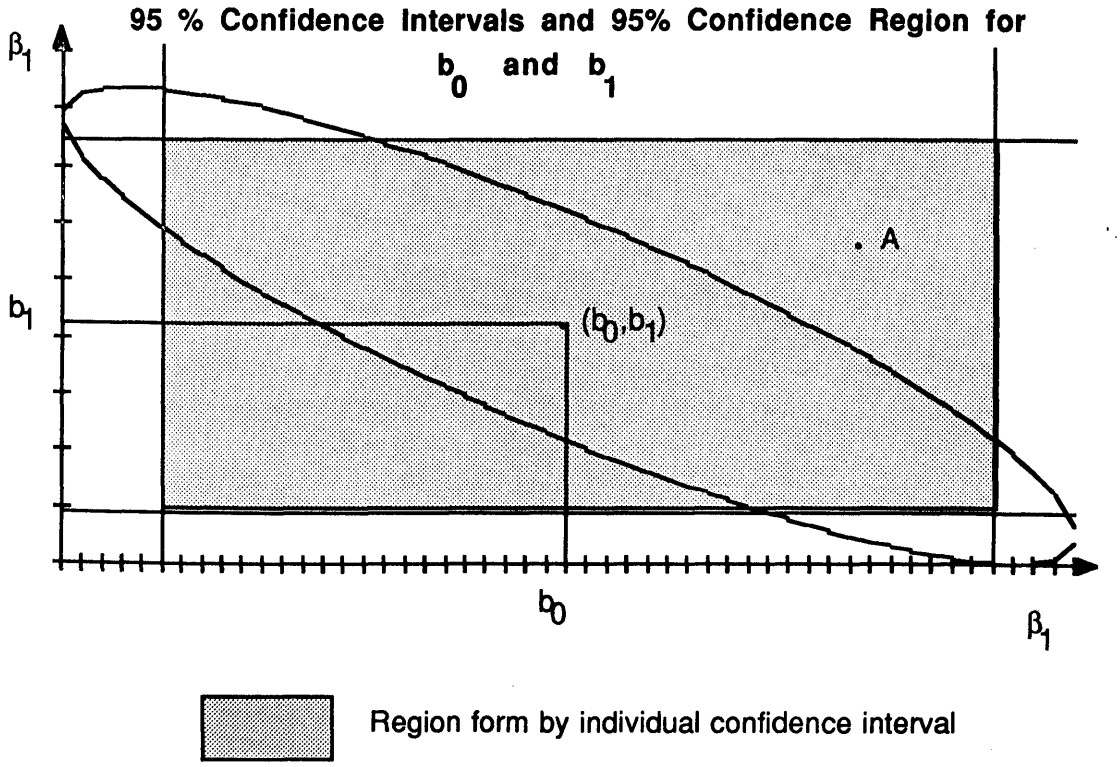

95 % Confidence Intervals and 95% Confidence Region for
b0 and b1
β1
b1
(b0,b1)
A
b0
β1
Region form by individual confidence interval

Table 2

$$Y = \mu + \sum_{i=1}^{3} a_i X_i + E \quad \text{Using Coding } X_i = \begin{cases} 1 \text{ if group } i \\ 0 \text{ otherwise} \end{cases}, \quad i=1, 2,$$

Group	X1	X2	X3	Yij - E	Interpretation
1	1	0	0	$\mu + a_1$	mean of group 1
2	0	1	0	$\mu + a_2$	mean of group 2
3	0	0	1	$\mu + a_3$	mean of group 3
4	0	0	0	μ	mean of group 4

$$Y = \mu + \sum_{i=1}^{3} a_i X_i + E \quad \text{Using Coding } X_i = \begin{cases} 1 \text{ if group } i \\ -1 \text{ if group n} \\ 0 \text{ otherwise} \end{cases}, \quad i=1, 2, \ldots 3$$

Group	X1	X2	X3	Yij-E	Interpretation
1	1	0	0	$\mu + a_1$	mean of group 1
2	0	1	0	$\mu + a_2$	mean of group 2
3	0	0	1	$\mu + a_3$	mean of group 3
4	-1	-1	-1	$\mu - a_1 - a_2 - a_3$	mean of group 4

Figure 3
One-Way Analysis of Variance Table

Source	df	SS	MSE=SS/df
Between Groups	K-1	$\sum_{i=1}^{k} J_i \, (\bar{Y}_i - \bar{Y})^2$	s_b^2
Within Groups	n - k	$\sum_{I}^{i=1} \sum_{Ji}^{j=1} (Yij - \hat{Y}_i)$	s_w^2
Mean	1	$n\bar{Y}^2$	
Total	n	$\sum_{I}^{i=1} \sum_{ji}^{j=1} Y_{ij}^2$	

J_i = number of measurements in the level

Figure 4

Layouts for Two-way ANOVA With An Equal Number of Observation in Each Cell

Row Factor	Column Factor 1	2	...	c	Row Means
1	$(Y_{111}, Y_{112}, \ldots, Y_{11n})$	$(Y_{121}, Y_{122}, \ldots, Y_{12n})$	...	$(Y_{1c1}, Y_{1c2}, \ldots, Y_{1cn})$	$\bar{Y}_{1.}$
2	$(Y_{211}, Y_{212}, \ldots, Y_{21n})$	$(Y_{221}, Y_{222}, \ldots, Y_{22n})$	...	$(Y_{2c1}, Y_{2c2}, \ldots, Y_{2cn})$	$\bar{Y}_{2.}$
...	...	...	...	...	...
r	$(Y_{r11}, Y_{r12}, \ldots, Y_{r1n})$	$(Y_{r21}, Y_{r22}, \ldots, Y_{r2n})$	...	$(Y_{rc1}, Y_{rc2}, \ldots, Y_{rcn})$	$\bar{Y}_{r.}$
Column Means	$\bar{Y}_{.1}$	$\bar{Y}_{.2}$	...	$\bar{Y}_{.c}$	$\bar{Y}_{..}$

Figure 5

The Mean Values of the Y for a Two-way ANOVA with Interactions

Factors

row	column	Y-E	Interpretation
1	1	$\mu + \alpha_1 + \beta_1 + \gamma_{11}$	The mean of all Y with Row factor equal to 1, Column factor equal 1, and the interaction between Row factor equal to 1 and Column factor equal 2.
1	2	$\mu + \alpha_1 + \beta_2 + \gamma_{12}$	The mean of all Y with Row factor equal to 1, Column factor equal 2, and the interaction between Row factor equal to 1 and Column factor equal 2.
1	3	$\mu + \alpha_2 - \beta_1 - \beta_2 - \gamma_{13}$	The mean of all Y with Row factor equal to 1, Column factor equal 3, and the interaction between Row factor equal to 1 and Column factor equal 3.
2	1	$\mu + \alpha_2 + \beta_1 + \gamma_{21}$	The mean of all Y with Row factor equal to 1, Column factor equal 1, and the interaction between Row factor equal to 2 and Column factor equal 2.
2	2	$\mu + \alpha_2 + \beta_2 + \gamma_{22}$	The mean of all Y with Row factor equal to 2, Column factor equal 2, and the interaction between Row factor equal to 2 and Column factor equal 2.
2	3	$\mu + \alpha_2 - \beta_1 - \beta_2 - \gamma_{23}$	The mean of all Y with Row factor equal to 2, Column factor equal 3, and the interaction between Row factor equal to 2 and Column factor equal 3.
3	1	$\mu + \alpha_2 + \beta_1 + \gamma_{31}$	The mean of all Y with Row factor equal to 3, Column factor equal 1, and the interaction between Row factor equal to 3 and Column factor equal 2.
3	2	$\mu + \alpha_3 + \beta_2 + \gamma_{32}$	The mean of all Y with Row factor equal to 3, Column factor equal 2, and the interaction between Row factor equal to 3 and Column factor equal 2.
3	3	$\mu + \alpha_3 - \beta_1 - \beta_2 - \gamma_{33}$	The mean of all Y with Row factor equal to 3, Column factor equal 3, and the interaction between Row factor equal to 3 and Column factor equal 3.
4	1	$\mu - \alpha_1 - \alpha_2 - \alpha_3 + \beta_1 - \gamma_{11} - \gamma_{21} - \gamma_{31}$	The mean of all Y with Row factor equal to 4, Column factor equal 1, and the interaction between Row factor equal to 4 and Column factor equal 2.
4	2	$\mu - \alpha_1 - \alpha_2 - \alpha_3 + \beta_2 - \gamma_{12} - \gamma_{22} - \gamma_{32}$	The mean of all Y with Row factor equal to 4, Column factor equal 2, and the interaction between Row factor equal to 4 and Column factor equal 2.
4	3	$\mu - \alpha_1 - \alpha_2 - \alpha_3 - \beta_1 - \beta_2 - \gamma_{11} - \gamma_{21} - \gamma_{31} - \gamma_{12} - \gamma_{22} - \gamma_{32}$	The mean of all Y with Row factor equal to 4, Column factor equal 3, and the interaction between Row factor equal to 4 and Column factor equal 3.

Figure 6

Two-way Analysis of Variance Table

Source	df	SS	MS =SS/df
Rows	I-1	$SSR = JK\sum_{i=1}^{I}(\bar{Y}_{i..} - \bar{Y})^2$	MSR
Columns	J-1	$SSC = IK\sum_{j=1}^{J}(\bar{Y}_{.j.} - \bar{Y})^2$	MSC
Interaction	(I-1)(J-1)	$SSRC = K\sum_{i=1}^{I}\sum_{j=1}^{J}(\bar{Y}_{ij.} - \bar{Y}_{.j.} - \bar{Y}_{.j.} + \bar{Y})^2$	MSRC
Residual	IJ(K-1)	SSE=TSS-SSR-SSC-SSRC	MSE
Correct total	IJK-1	$TSS = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}(Y_{ijk} - \bar{Y})^2$	

Possible Series of Test Statistics

Factor	Type of Factor	Ho	Test Statistic
Row	Fixed	$H_0: \alpha_1 = \alpha_2 .... = \alpha_I = 0$	MSR/MSE
Column	Fixed	$H_0: \beta_1 = \beta_2 .... = \beta_J = 0$	MSC/MSE
Interaction	Fixed	$H_0: \gamma_{11} = \gamma_{21} .... = \gamma_{IJ} = 0$	MSC/MSE
Row	Fixed	$H_0: \alpha_1 = \alpha_2 .... = \alpha_I = 0$	MSR/MSE
Column	Random	$H_0: \sigma_{1c} = \sigma_{2c} .... = \sigma_{Jc} = 0$	MSC/MSRC
Interaction	Fixed	$H_0: \gamma_{11} = \gamma_{21} .... = \gamma_{IJ} = 0$	MSC/MSE
Row	Random	$H_0: \sigma_{1r} = \sigma_{2r} .... = \sigma_{Ir} = 0$	MSR/MSRC
Column	Fixed	$H_0: \beta_1 = \beta_2 .... = \beta_J = 0$	MSC/MSE
Interaction	Fixed	$H_0: \gamma_{11} = \gamma_{21} .... = \gamma_{IJ} = 0$	MSC/MSE
Row	Random	$H_0: \sigma_{1r} = \sigma_{2r} .... = \sigma_{Ir} = 0$	MSR/MSRC
Column	Random	$H_0: \sigma_{1c} = \sigma_{2c} .... = \sigma_{Jc} = 0$	MSC/MSRC
Interaction	Fixed	$H_0: \gamma_{11} = \gamma_{21} .... = \gamma_{IJ} = 0$	MSC/MSE

HIGH PERFORMANCE I/O SUBSYSTEMS

Anthony G. Mungal.

ABSTRACT

It is readily apparent that major breakthroughs in technology coupled with the evolving and shifting trends of the data processing industry will necessitate revolutionary methodologies for the configuration and classification of data. As the quantity of on-line data increases, its management becomes a primary concern. Incorporated with that management concern are the issues of how to solve the ever growing need for capacity and the incessant demand for better performance.

Data access patterns in many environments have proven that a significant amount of access activity tends to reference a small percentage of the available data. This can lead to some rather peculiar circumstances. The well known phenomena of DASD skew comes to mind as an example of such a case, whereby access is required to different portions of data spanning different time patterns. How can one deal with these various phenomena?

This tutorial will deal with the issues surrounding the isolation, identification, and subsequent configuration of high performance data. A thorough understanding and definition of high performance data, and how it should be configured for capacity and performance reasons, can solve the major problems of most I/O environments.

It is also intended to address the data classification issues of high performance data and provide some general guidelines as to the various storage product classes that may be required to alleviate performance and capacity problems. One primary focus is in emphasizing concepts that are apparent with high performance data; for example, the concept of access density and its interpretation. Tools and techniques that emphasize the ease of analysis of high performance data and the on-going performance monitoring of it are also discussed.

Finally, the hierarchical structure of the storage system is examined. That leads to a discussion of the various costs and capacity type arguments that are of paramount concern to today's discriminating user, and provides a frame work whereby one can analyze the effects of increasing capacity on data centre costs.

INTRODUCTION

Much of the material within this paper was derived from industry wide concerns as to the disposition of high performance data. The issues centered around the configurational aspects of data, the current and future technologies employed, the capacities, the reliability criteria, the availability criteria and, of course, cost. In looking at high performance subsystems, it is then proper to consider technology offerings that serve the needs of high performance data both current and future.

- TODAY'S PROBLEMS
- TOMORROW'S TECHNOLOGY
- DEFINITION OF HIGH PERFORMANCE DATA (HPD)
- ACCESS DENSITY
- CAPACITY OR PERFORMANCE
- APPLICABILITY OF CACHE AND SSD TO HPD
- BUSINESS CASE STRATEGY

FIGURE 1 - OUTLINE

This paper (see Figure 1) will deal with the problems that are apparent within todays environment; it will then speculate on how the technology of tomorrow (either rumored or real) could impact those problems; it will define high performance data and proceed to isolate methods for analyzing high performance data, and finally deal with certain configurational aspects of it. The application of cache and solid state technologies along with their associated business case strategies are also examined.

PROBLEMS? . . . FIRST CONSIDER

- CONTINUING TRENDS (PROCESSING)
- TECHNOLOGY TRENDS (ARCHITECTURE)
- PRODUCTIVITY TRENDS (HUMAN)
- APPLICATION TRENDS (BUSINESS)

LET'S CONSIDER EACH IN TURN

FIGURE 2 - FOUR PERSPECTIVES ON TRENDS

TODAY'S PROBLEMS

In trying to define the problems (see Figure 2) apparent in todays environment, it is perhaps best to look at four perspectives on trends:

the continuing trend (information processing), the technology trends (architecture), productivity trends (human), and the application trend (business).

The information processing trend (see Figure 3) has been, and continues to be, the shift from batch to online services; that is not to say that batch is decreasing, instead, it is ever increasing. The true recognizable trend is that more and more of a company's business is moving to online services. This has some interesting ramifications. The first is that the workloads within the computing complex is increasing in a non-linear format. This near exponential growth requires more online capacity (storage) as more applications are implemented. The tasks associated with the backup of active data and the migration of inactive data becomes compounded, hence storage management principles and practices need to be revised and updated. As a matter of fact, many user lament the fact that today they are adding storage capacity in units of gigabytes instead of megabytes as was the case only a few years ago; this punctuates the need for more effective storage management policies. Also, the need for high availability and reliability of the various systems is of paramount concern.

The technology trends (see Figure 4) have enabled DASD capacity to increase by a factor of sixty, CPU speeds by a factor of thirty and DASD performance by a mere factor of four; this defines the biggest problem in I/O subsystems today. The ever diverging gap of CPU speeds and DASD performance is a constant source of embarrassment to major hardware vendors, and a nagging headache to the performance analyst. In fact, Singh et al [8] commented "that during the design phase of the 3090, analytical evaluations showed that additional improvements in the I/O subsystem performance would be of considerable importance for a processor in that performance range". They continue to comment that cached DASD subsystems would offer the much-needed improvement in DASD service times.

The productivity trend (see Figure 5) is based on the human need for fast (low), and consistent response times. Thadhani [2] and others studied the relationships of productivity and response time. It was concluded that subsecond response times provided the highest productivity gains. In most environments today, subsecond response time is a fact of life, and an expectation; the only exception being those application areas that involve a considerable amount of data entry and subsequent processing.

The application trend depicts the increasing use of fourth or even fifth generation languages for application implementation. The impact on the I/O subsytem is manifested by way of large databases and many physical I/Os per transaction; this makes DASD tuning a challenge for even the most experienced performance analyst. This trend is expected to continue in an even more accentuated manner, suggesting that at some point in the future significant changes in the I/O architectures will be required to meet the response times, the availability, and reliability criteria.

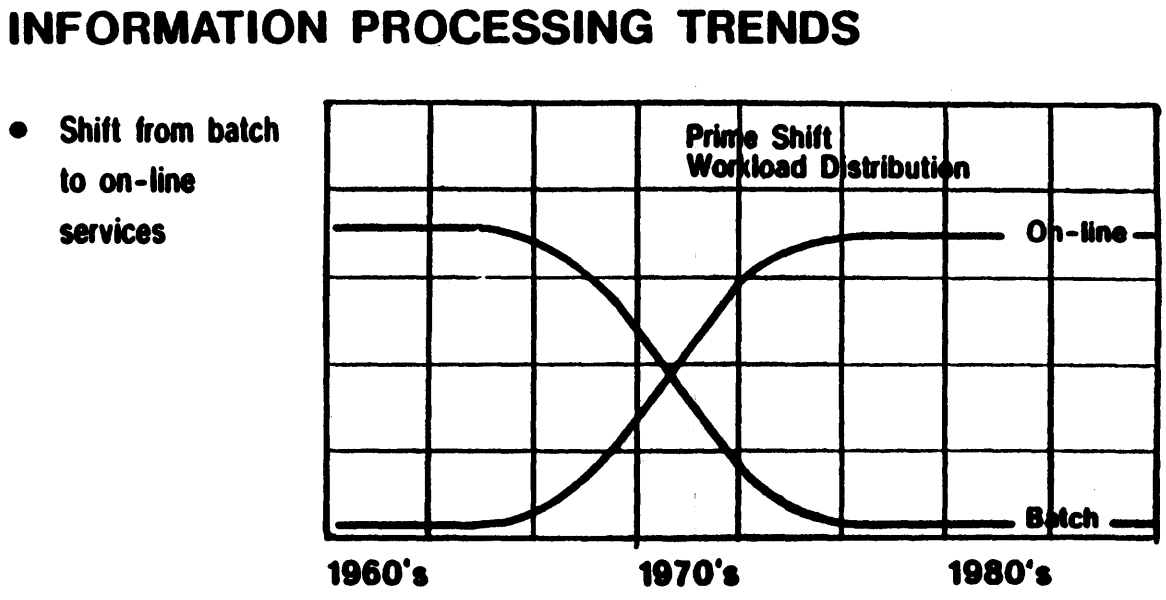
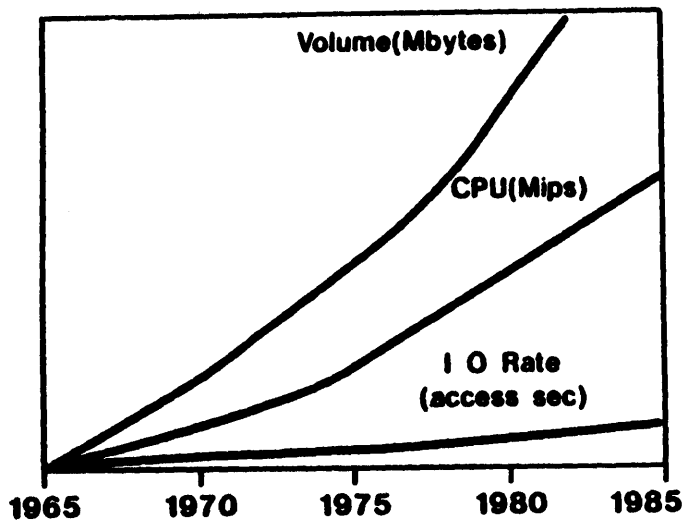


FIGURE 3 - INCREASING PRIORITY ON ON-LINE SERVICES

TECHNOLOGY TRENDS



DASD capacity, 60X larger

CPU speed, 30X faster

DASD performance, 4X faster

FIGURE 4 - COMPARING THE TECHNOLOGIES: CPU & DASD.

PRODUCTIVITY AS A FUNCTION OF RESPONSE TIME

- **Productivity Factors**
  - **Fast response**
  - **Consistent response**

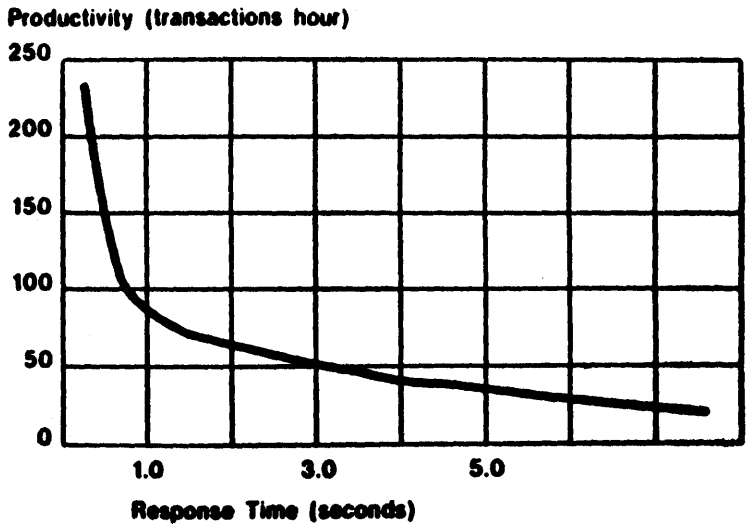


FIGURE 5 - THE PRODUCTIVITY TREND: SUBSECOND RESPONSE TIME.

Such changes may include device transparency and increased I/O parrallelism.

The trends therefore highlight the need for: increased availability and reliability, increased storage capacity, better storage management, increased I/O capacity, and better I/O service time.

It is not surprising then that approximately seventy-five percent of all reported performance problems are due to I/O contention. This is precisely why the technologies of I/O subsystems need to be evaluated on a continuing basis.

TOMORROW'S TECHNOLOGY ?

- BIGGER/FASTER PROCESSORS
- MORE CHANNELS/ FASTER CHANNELS
- EXPANDED STORAGE
- HIGHER CAPACITY DISKS (3X, 4X)

FIGURE 6 - TOMORROW'S TECHNOLOGY ??

FUTURE PROCESSORS

It is evident (see Figure 6) that future processors will be much bigger and faster; this will only widen the gap between processor speed and DASD access times, thus making an already bad situation worse. It is intuitively obvious that the usage of those processors will require more sprawling I/O subsystems. The key fundamental metric is data bandwidth; this increases as a function of processor speed. It is also a fact that future processors will offer more channels that may be even faster than existing ones; this provides the ability to address the I/O bandwidth requirement, but it also implies the procurement of newer technologies that will attach to it. One can speculate on such technologies as the triple or quad density disk drives and the availability of 4.5 and/or 6 megabyte transfer rates.

FASTER CHANNELS

Even if higher transfer rates were to become a reality with existing technology, it only still addresses one component of the I/O service time - that being transfer time. For example, (see Figure 7) consider a 4K transfer on a 3 megabyte channel taking approximately 1.3 milliseconds and then the same block on a 6 megabyte channel halving that time to 0.65 milliseconds. The saving of 0.65 milliseconds per I/O for a processor doing 400 I/Os per second will be approximately 260 milliseconds per second. Now consider one quarter of 400 I/Os (100 I/Os) with a current service time of 16 milliseconds which is later reduced by a mere 3 milliseconds to 13 milliseconds, thereby yielding a savings of 300 milliseconds per second. It is obvious that treating even a subset of the total I/O rate to a lower service time can outperform transferring the whole set of I/Os at an increased transfer rate. The above scenario can be computed for a 4.5 megabyte transfer rate, and is left as an exercise for the reader.

DO YOU WANT FASTER CHANNELS?

3 MB/SEC 4K BLOCK = 1.30 MS

4.5 MB/SEC 4K BLOCK = 0.87 MS

6 MB/SEC 4K BLOCK = 0.65 MS

- 400 I/O PER SEC @ 6 MB/SEC SAVES 260 MS/SEC

- CONSIDER: 100 I/O PER SEC WITH SERVICE TIME OF 16 MS REDUCED TO 13 MS
  ==> SAVINGS OF 300 MS

NOTE: SEEK AND LATENCY ARE NOT IMPACTED
THEY REMAIN A FUNCTION OF THE TECHNOLOGY

REDUCING THE TRANSFER TIME EMPHASISES THE NEGATIVE EFFECTS OF SEEK AND LATENCY

FIGURE 7 - CHANNEL TRANSFER RATE ANALYSIS

In the case of DASD I/O (see Figure 8), seek and latency are not influenced; and they remain a function of the technology. The percentage improvement of the I/O in going from 3 mb/sec to 4.5 mb/sec was 12.2; from 4.5 mb/sec to 6 mb/sec, it was 6.25; and from 3 mb/sec to 6 mb/sec, it was 17.7. Duhl [12] discusses many of the issues surrounding faster channels and their applicability to future subsystems. Thus, faster channels alone is not a solution to improving I/O subsystem performance; one must impact the negative effects of seek and latency.

TYPICAL 4K I/O

	3 MB CHANNEL		4.5 MB CHANNEL		6 MB CHANNEL	
SEEK	3.0	17.9%	3.0	20.4%	3.0	21.7%
LATENCY	8.3	49.4%	8.3	56.4%	8.3	59.7%
RPS(20%)	4.18	25.0%	2.56	17.4%	1.86	13.7%
XFER 4K	1.3	7.7%	0.87	5.9%	0.65	4.7%
	16.8	100	14.73	100	13.21	100

FASTER CHANNELS ALONE IS NOT A SOLUTION TO IMPROVING I/O SUBSYSTEM PERFORMANCE ===> MUST REDUCE THE EFFECTS OF SEEK AND LATENCY

ASSUMPTIONS: '80 CLASS DASD; MINIMUM SEEK TIME
INTERNAL PATH BUSY : 20% FOR 3 MB
13.3% FOR 4.5 MB
10% FOR 6 MB
10% FOR 6 MB

FIGURE 8 - INCREASING TRANSFER RATE ON DASD I/O

EXPANDED STORAGE

The current uses of expanded storage on the 3090 processors is an effective means of reducing page and swap traffic. While other uses exist for expanded storage (data base buffers, VIO, and linear data sets), cost, volatility and sharing still remain as outstanding concerns. There are other ways to provide better cost performance for any given system; some of these issues are addressed in detail by Mungal [11].

HIGHER CAPACITY DISK

Higher capacity disk (rumoured 3X & 4X - 3 and 4 times the current "D" capacity respectively) holds great promise for the realization of a much lower than current cost per megabyte, and associated environmentals. However, even that promise degrades to superficiality when one scrutinizes the access density characteristic of them. Access density is discussed later in this paper.

CHARACTERIZING HIGH PERFORMANCE DATA

HIGH PERFORMANCE DATA IS CHARACTERIZED BY:

- HIGH I/O RATES
- LOW SERVICE TIME REQUIREMENT
- HIGH AVAILABILITY REQUIREMENT

FIGURE 9 - CHARACTERIZING HIGH PERFORMANCE DATA

High performance data (see Figure 9) is characterized by high I/O rates, a low service time requirement, and high availability requirement. In fact, the high availability criteria led to the development of the dual frame architecture featured in current DASD controllers. There are many questions (see Figure 10) to be answered in further qualifying high performance data. The first category of questions qualify the "what is" high performance data, i.e., what subsystems utilize it, what are the characteristics of it in terms of read to write ratio, I/O rates and so on.

CHARACTERIZING/QUALIFYING HIGH PERFORMANCE DATA

- WHAT COMPRISES YOUR HIGH PERFORMANCE DATA (HPD) ?
- WHAT SUBSYSTEMS REQUIRE ACCESS TO HPD ?
- WHAT ARE THE CHARACTERISTICS OF IT ? I.E. R:W RATIOS, I/O RATES ETC
- WHAT WILL YOU DO IF THE DEMAND FOR IT INCREASES ?
- WHERE DOES IT CURRENTLY RESIDE ?
- WHERE SHOULD IT BE PLACED IN THE FUTURE ?
- HOW MUCH EFFORT IS REQUIRED TO MONITOR ITS PERFORMANCE AND RESPONSE / SERVICE TIMES ?
- HOW MUCH DOES IT COST ?
- HOW MUCH OF MY BUSINESS DEPENDS ON IT ?
- HOW MUCH AVAILABILITY IS ACCEPTABLE ?

FIGURE 10 - FURTHER CHARACTERIZING/QUALIFYING HIGH PERFORMANCE DATA

The second category of questions qualify the "where" high performance data should reside, i.e.,on the processor (1 or more), disk, solid state or some combination thereof.

Improper placement/configuration of high performance data will translate to increased cost to the user in terms of dollars, performance, availability, and even reliability

The last category of questions is concerned with qualifying "how much" effort and capital is invested on an ongoing basis in high performance data. This category of questions defines the reliability and availability criteria since they are a direct function of cost; it also deals with the cost per megabyte issue.

ACCESS DENSITY

The concept of access density is a well understood one. It is defined as accesses per second per gigabyte; and as yet, serves as the only correlation between access activity to a volume (I/O rate) and volume capacity (gigabytes). In fact, access density has declined steadily over the years with a promise for even further decline if higher capacity devices become a reality. Arresting this steady decline can only be addressed by accommodating more actuators per volume in future dasd devices.

Figure 11 shows the access density for "D" type DASD boxes; note the rapid decrease of access density as i) the string length is varied from 1 to 4 boxes, and ii) the I/O rates is varied from 20 to 100 I/O's per second. In fact, one may well argue that I/O rates in excess of 60 I/O's per second to a string of D's requires cache, and hence, this data may be so interpreted.

Recent GUIDE surveys indicated that the average allocation percentage across strings in many large environments was only 45%. Figure 12 shows the behaviour of access density when only 45% of the string is allocated/used; again a range of I/O rates is considered. This suggests that users traded capacity to achieve performance as manifested by the greatly increased access density (compared with Figure 11).

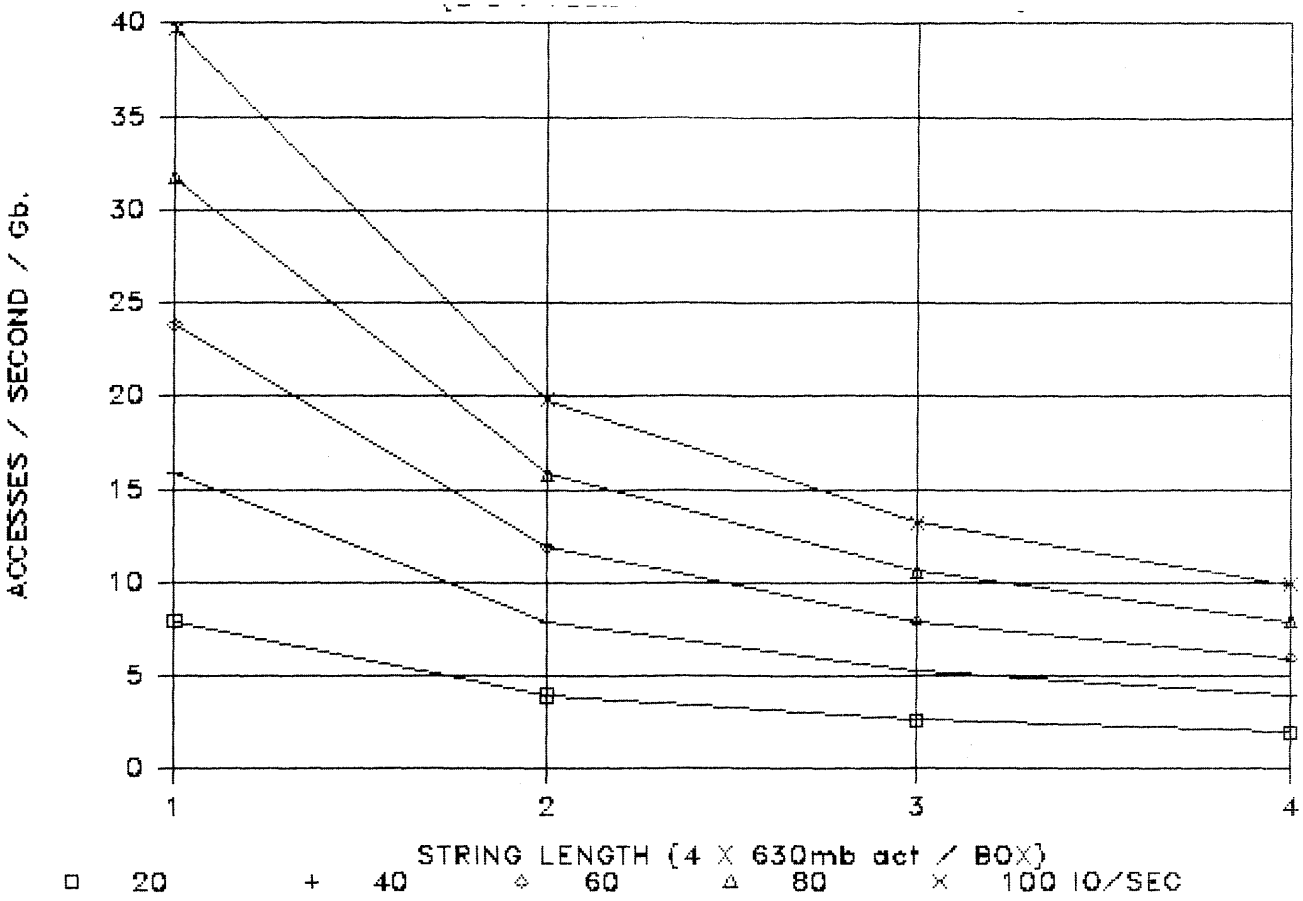


FIGURE 11 - ACCESS DENSITY VS STRING LENGTH (100% ALLOCATED): A COMPARISON OF 5 I/O RATES.

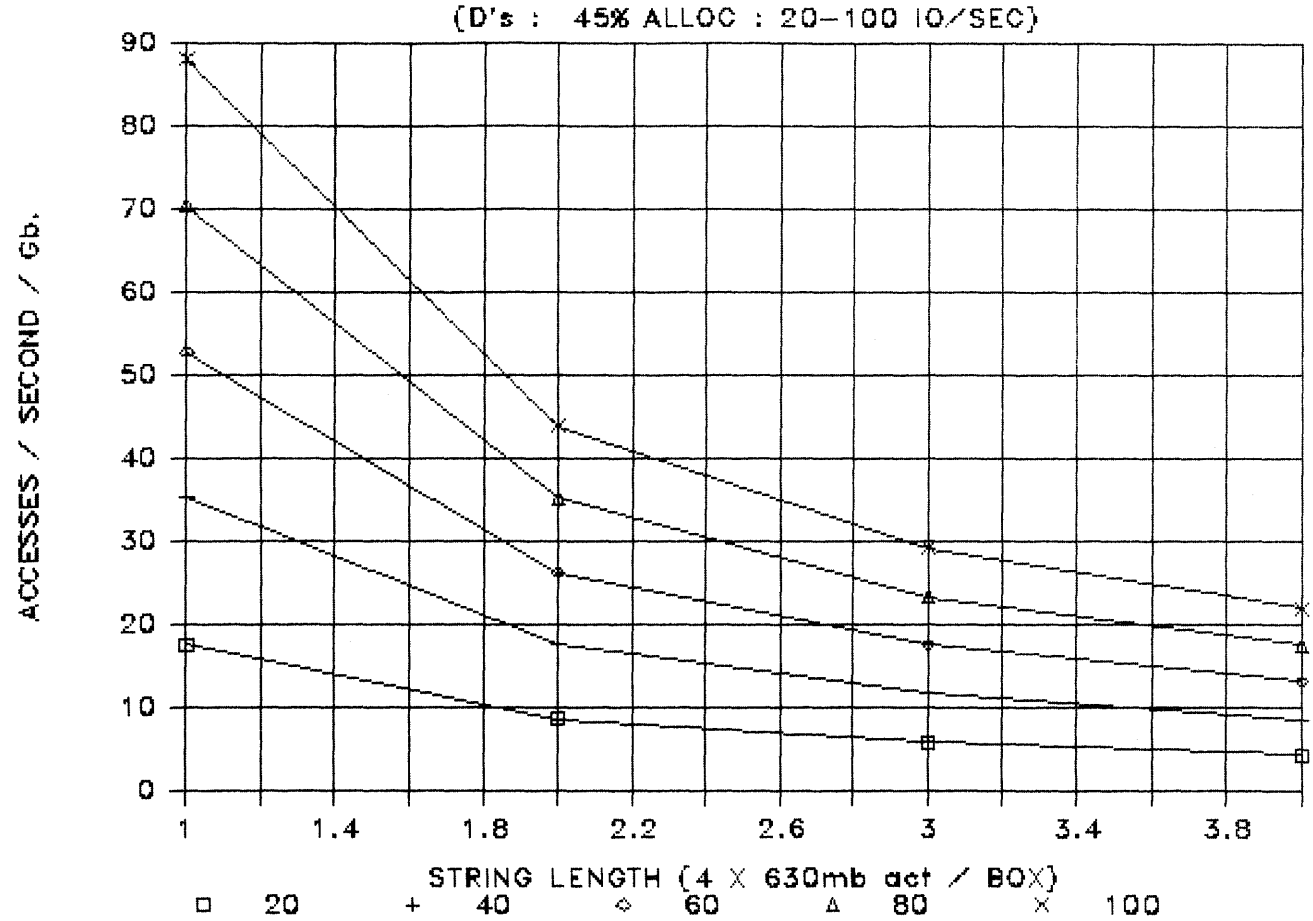


FIGURE 12 - ACCESS DENSITY VS STRING LENGTH (45% ALLOCATED): A COMPARISON OF 5 I/O RATES.

Figure 13 shows similar data to Figure 12, except that it demonstrates the behaviour of access density as the percentage allocation is increased (string becomes loaded), and the string length (1 to 4 boxes) is varied. Given that access density varies with any/all of I/O rates, string length, and percent allocation, suggests that the job of capacity planning and/or tuning is somewhat simplified; since it is usually easy to hold at least one of them constant (e.g. string length).

Figure 14 depicts the relationship of access density with I/O rate, when the latter is subjected to a shallow exponential growth. This sort of analysis is useful for determining the "strength" of the subsystem, and it's ability to respond to wide fluctuations in I/O load; this strength is typified by the solid state disk.

Figure 15 shows the behaviour of access density and gigabytes per string; the I/O rates vary from 40 to 160 I/O's per second. Even with the higher I/O rates (> 80), it becomes evident that beyond a certain string capacity, one must be prepared to employ cache as an effective means of achieving acceptable access densities. For example, 20 gigabytes of data (2 full strings of E's) can be made to perform acceptably if sufficiently high I/O rates can be sustained; this is the principal domain of caching.

In general, similar analysis for any/all DASD device types is easy to accomplish. The message from such analyses is only to suggest that prior knowledge of criteria such as I/O rates, quantity of data to be accessed, and device type can go a long way in assisting the configuration/tuning efforts; cost emerges from this as being optimized.

CAPACITY OR PERFORMANCE

McNutt [1] used data from a survey of fifty-eight large installations (see Figure 16) spanning more than nine thousand DASD devices. This shows that a high percentage of the devices are less than ten to fifteen percent utilized; this subset qualifies for double capacity pairing . Only a small amount of the devices exhibit high busyness; this subset of devices is the classic candidate for residence on single capacity actuators, cache and/or solid state disk. This is an example of a methodology that would allow the identification of high performance data. An interesting by-product of such a methodology is the isolation of other data.i.e. data/string homogeneity. Another methodology would be that of employing access density as previously discussed.

CONFIGURATIONAL CONSIDERATIONS (SINGLE AND DOUBLE CAPACITY)

The strategy of placing data on double capacity devices looks at the device busyness which in turn is related to the number of concurrently open data sets per volume. On these devices, the potential exists for significantly more data sets , thereby increasing the odds for

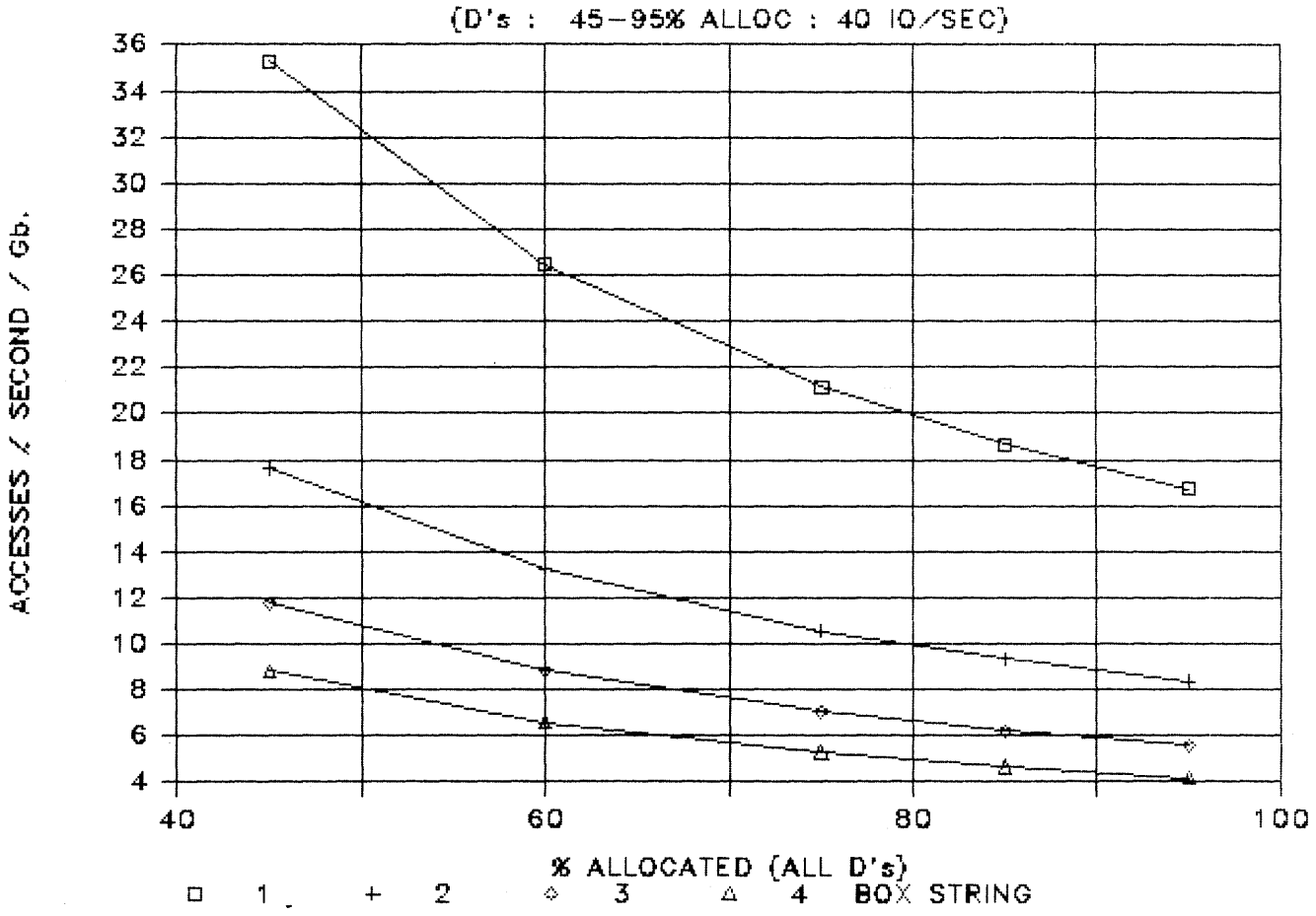


FIGURE 13 - ACCESS DENSITY VS PERCENT ALLOCATION:
A COMPARISON OF STRING LENGTHS

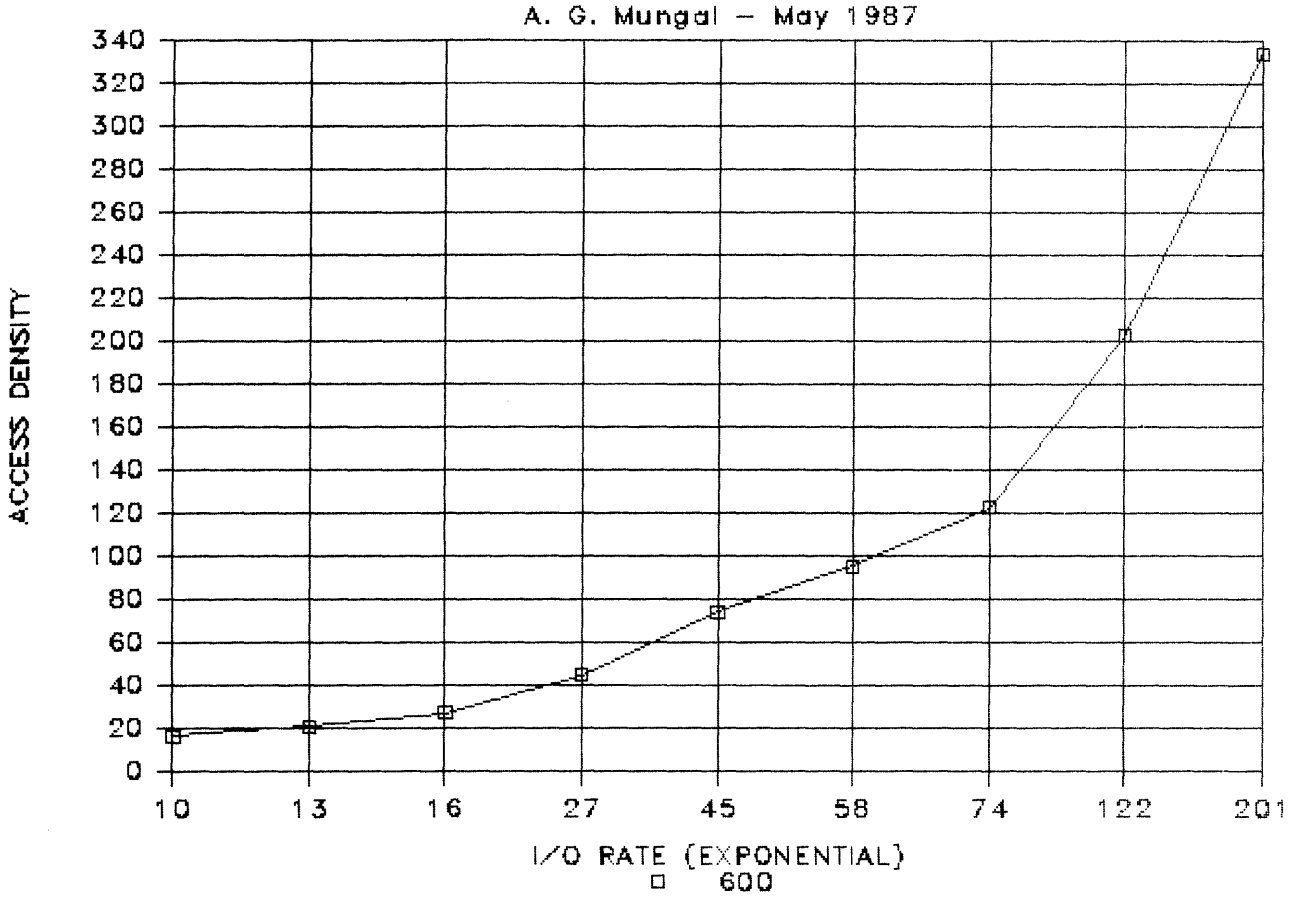


FIGURE 14 - ACCESS DENSITY VS EXPONENTIAL I/O RATE:
EXAMINING I/O SUBSYSTEM STRENGTH.

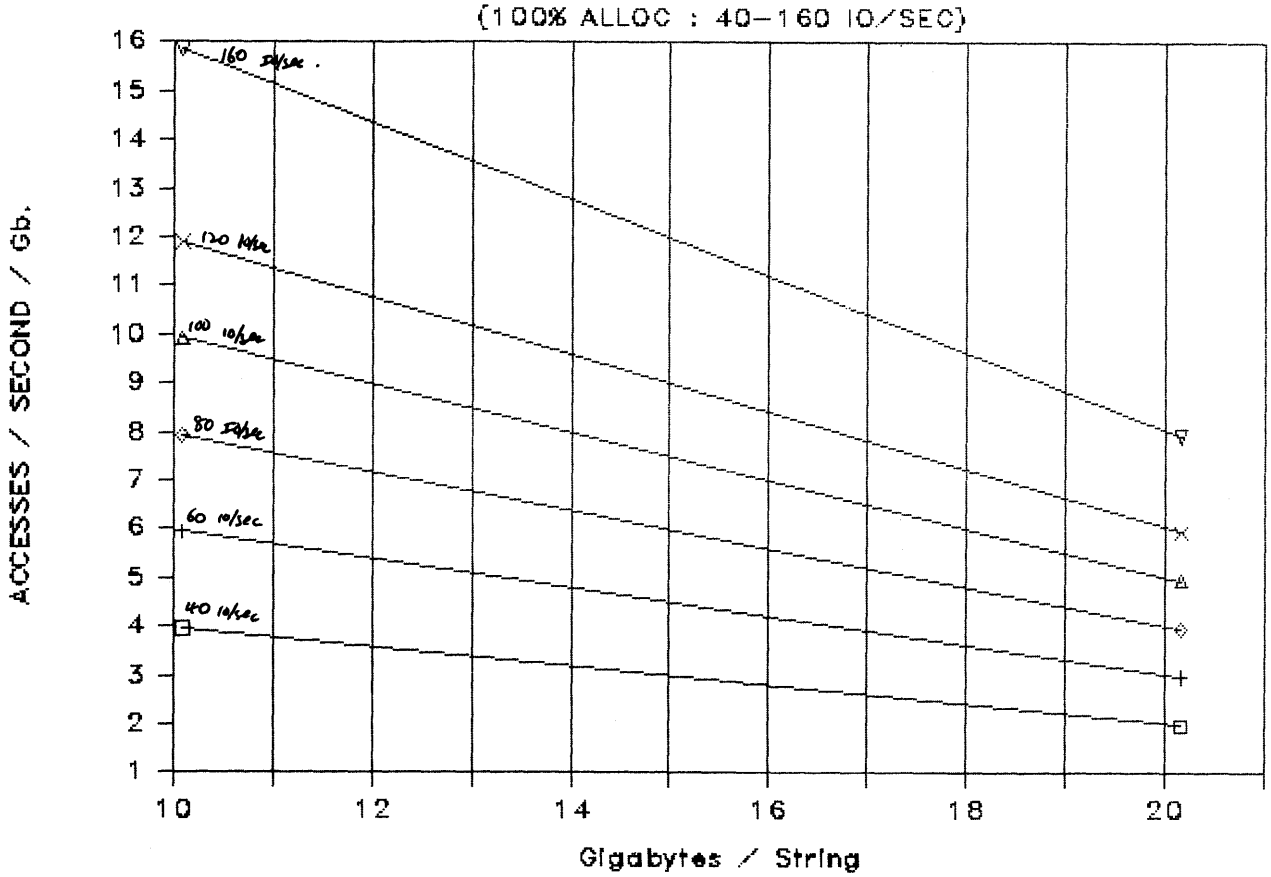


FIGURE 15 - ACCESS DENSITY VS GIGABYTES PER SUBSYSTEM:
A LOOK AT CACHED AND UNCACHED I/O RATES.

▶ IBM MEASURED DATA - 58 INSTALLATIONS
- 9130 DEVICES
- 3350/3380

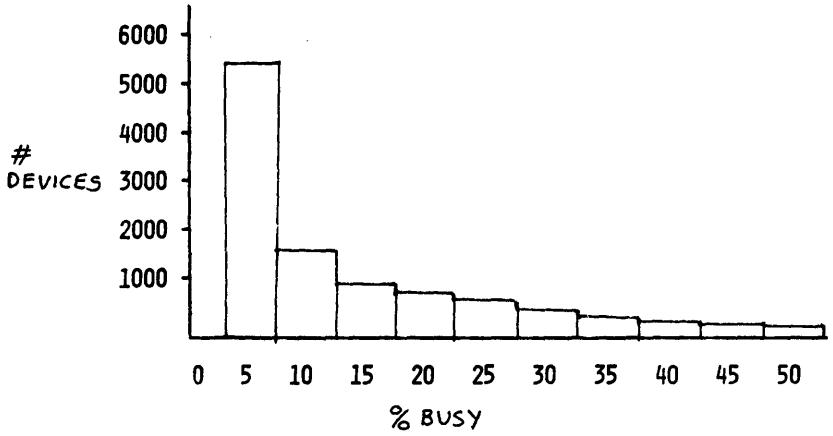


FIGURE 16 - A SURVEY OF DEVICE BUSY.

contention which in turn would result in degraded I/O times.

RATIONALE FOR ACQUIRING "D" TYPE DASD

- HIGHER ACCESS DENSITY THAN "E" TYPE DASD
- UNCACHED: VERY GOOD PERFORMANCE/CAPACITY
- CACHED: EXCELLENT PERFORMANCE
  INCREASED ALLOCATED CAPACITY
  INCREASED ACCESS DENSITY

FIGURE 17 - APPLICABILITY OF "D" TYPE DASD

Many customers (see Figure 17) share the concern of whether or not single capacity devices are still applicable to their environment. In fact, many high busy volumes are perfectly suited to single capacity devices; these devices provide a good blend of capacity and performance. They also have a higher access density than their double capacity counterparts. By caching single capacity devices, one can boost both capacity and performance , which in turn results in even better access density.

How much intermix of single and double capacity should be tolerated at the string level? The answer to this question lies within an examination of the access density required for the data that is to be configured. This can be determined from Resource Management Facility data (RMF).

Figure 18 depicts the performance considerations for single capacity disks according to string length. It suggests the service times for certain I/O rates for any given number of actuators (up to 16).

PERFORMANCE CONSIDERATIONS

SIO/SEC/ ACTUATOR	A	B	C	D
5	14.2	15.7	16.3	16.9
10	16.5	17.9	19.5	21.4
15	18.2	20.9	24.4	29.2

A = AD4 *** MODELED DATA
B = AD4 + BD4
C = AD4 + BD4 + BD4
D = AD4 + BD4 + BD4 + BD4

(4K BLOCKS; M/G/m)

FIGURE 18 - SUBSYSTEM RESPONSE TIMES AND STRING LENGTH

Having dealt with the issues surrounding single and double devices, the discussion shifts to true high performance data.

HIGH PERFORMANCE SUBSYSTEMS

High performance data (see Figure 19) belongs on high performance subsystems. When configured as such, the benefits provided will at least include the following: low service times, better I/O load, better I/O capacity, less tuning effort, consistency of service times (and the associated response times) and lower data centre costs. This leads to an examination of the benefits associated with caching.

HIGH PERFORMANCE DATA BELONG
ON A HIGH PERFORMANCE SUBSYSTEM

WHY ??

- SERVICE TIMES
- I/O LOADING ENHANCED
- BETTER USE OF EXISTING CAPACITY
- LESS TUNING EFFORT
- CONSISTENCY (TUNING; RESPONSE/SERVICE TIMES)
- FINANCIALLY ATTRACTIVE

FIGURE 19 - BENEFITS OF CONFIGURING HPD SUBSYSTEMS

CACHING

Before one can proceed with caching ,it would be useful to digress on cache analysis as a method of determining the cacheable data within any environment. Mungal [6] deals with this in detail. Other references include Gray [9] and Friedman [10]. Cache analysis is not dealt with at this time.

In applying cache to any environment (see Figure 20), one can adopt either or both of the following methodologies: dedicated caching and selective caching.

- DEDICATED CACHING
- SELECTIVE CACHING

ANALYSIS WILL DETERMINE WHICH ONE OR WHICH BLEND IS MOST APPLICABLE

FIGURE 20 - CACHING METHODOLOGIES

DEDICATED CACHING

Dedicated caching requires a more detailed data analysis of the environment. Such analysis will identify all of the cacheable volumes and will suggest which ones are best suited to a sufficiently large cache; this subsystem contains only cached data, since all volumes are defined as cacheable. This cache methodology has the advantages of being very stable, in that I/O tuning is minimized since the data is static by default.

Dedicated caching also lends itself to data homogeneity since most, if not all, of those volumes in the subsystem will exhibit similar read to write ratios, hit ratios and I/O rates. It is financially very attractive since it has the ability to provide the highest access density, thereby minimizing effective cost per megabyte.

The only major disadvantages with dedicated caching is that only a finite amount of data within any environment qualifies and data homogeneity is sometimes difficult to attain and even more difficult to maintain.

SELECTIVE CACHING

Selective caching identifies the hot data within a given subsystem; only that data is defined as cacheable to a sufficiently large cache and the subsystem is monitored from time to time. It is already obvious, from witnessing various customer environments that selective caching is the preferred means of deploying cache controllers to handle the needs of high performance data. The cache size is suitable to the amount of data, the I/O rates, and locality of reference patterns for the cacheable volumes within the subsystem, hence the economic benefits are maximized.

The obvious benefit of selective caching is that it allows a unique starting point for cache on a per string basis, thereby providing the very flexible growth capabilities in terms of the cache size, I/O rates and device capacities. Selective caching provides a very attractive cost/performance ratio.

Dataset level caching allows a significantly better level of granularity for dealing with dasd skew. This is a major enhancement to the overall process of selective caching.

Figure 21 shows the relationship between increasing I/O rates and response times for both cache and uncached subsystems. It is evident that cache subsystems provide the user with the ability to increase the I/O load at the same or lower response time that an uncached system, thereby gaining I/O capacity; or decrease the response time for the same load as an uncached subsystem, thereby achieving better response time. Those benefits are important to the capacity planner since they reflect the ability of the cache controller to behave as multiple uncached controllers.

Figure 22 examines a sample business case which serves to justify the cost of caching against any buyback in allocated capacity on either a volume or string level. In Figure 22, line A represents the capital cost of an uncached controller; line B represents an adjusted cost which reflects the level of allocation within that string (i.e., industry surveys indicate that strings are somewhere around forty five percent allocated);line C indicates the incremental cost of adding cache to this configuration, and thus has an increased cost per megabyte associated with it; line D represents the new cost per megabyte associated with being able to utilize more of the disk capacity once caching is activated; the difference between lines B and D is representative of the cost savings once a cache is deployed. The sample cost per megabyte for each curve (A thru D) is evidenced in Figure 23.

In summarizing the use of cache technology as it applies to todays high performance subsystems the following benefits are obvious: better device utilization of the cached actuators, lower service times, increased I/O capacity at the cached device and subsystem level, incremental savings in both floor space

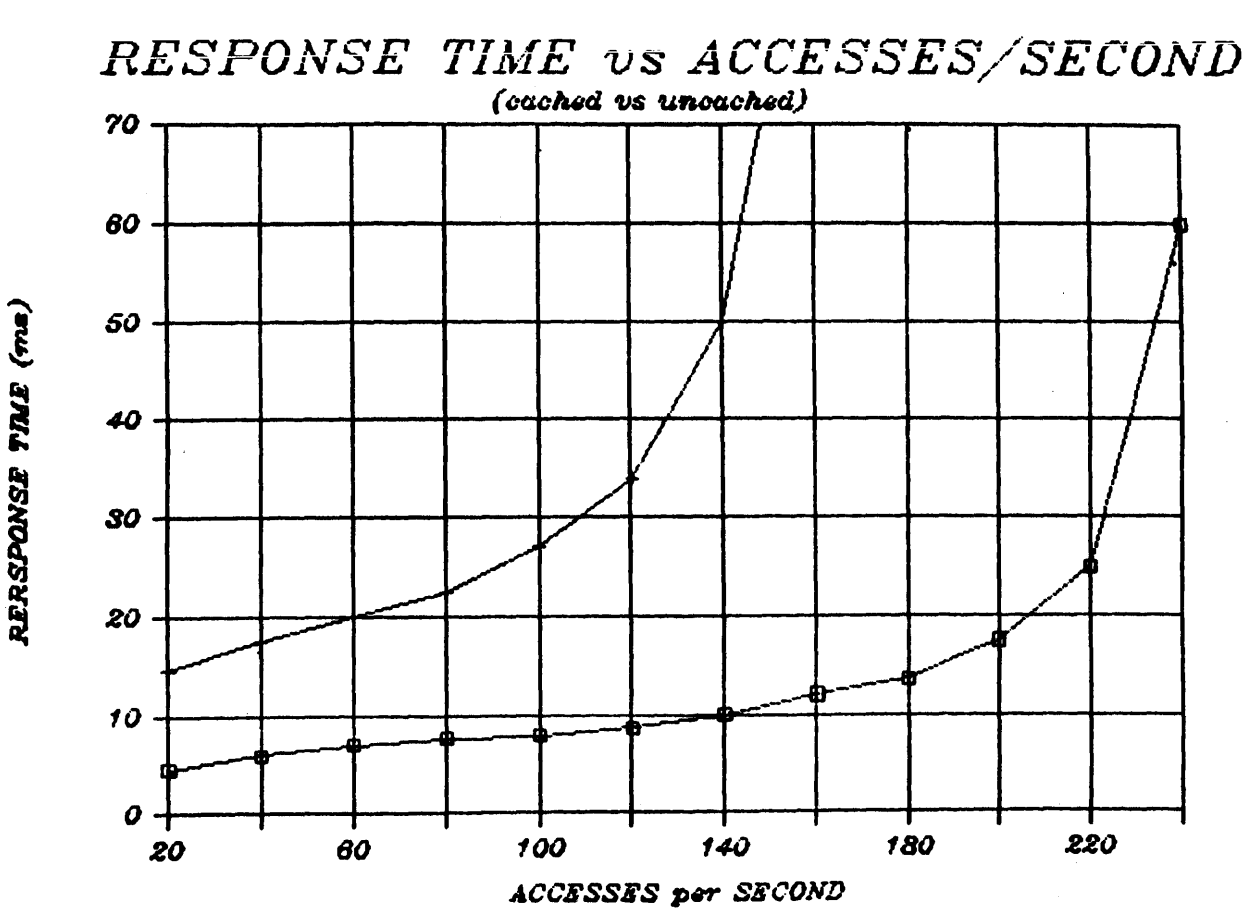


FIGURE 21 - RESPONSE TIME VS ACCESSES / SECOND:
COMPARING CACHED AND UNCACHED SUBSYSTEMS.

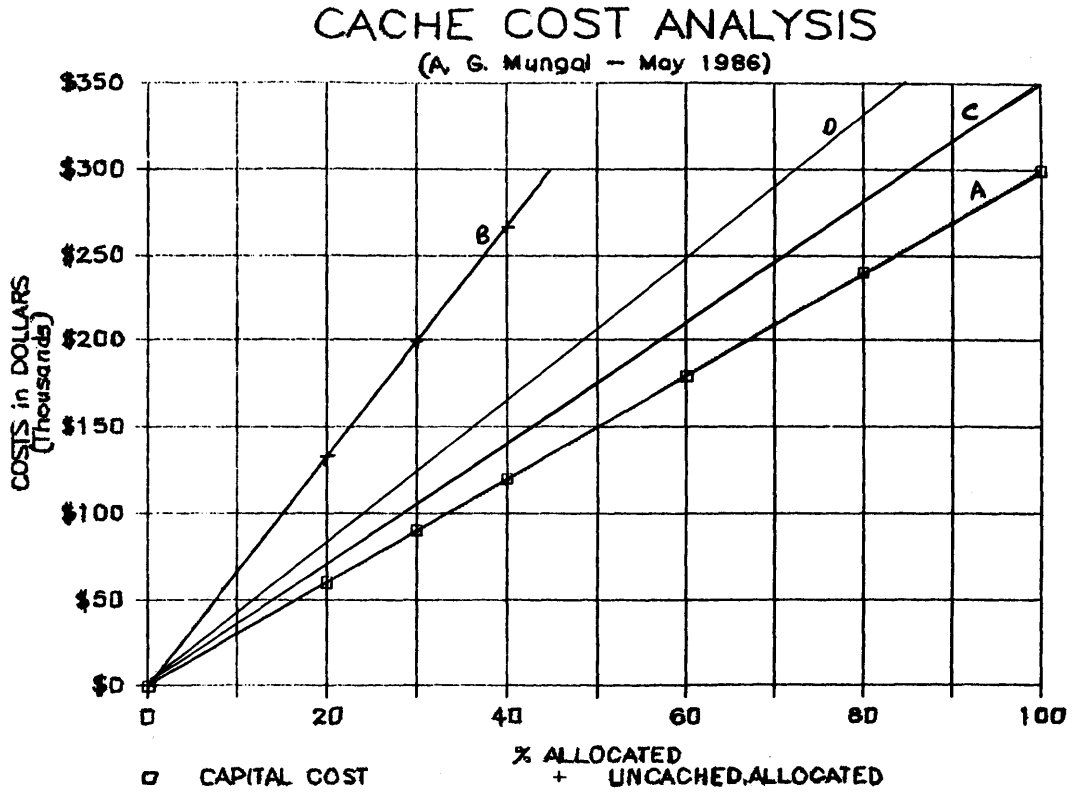


FIGURE 22 - A BUSINESS CASE JUSTIFICATION FOR CACHE: SAMPLE.

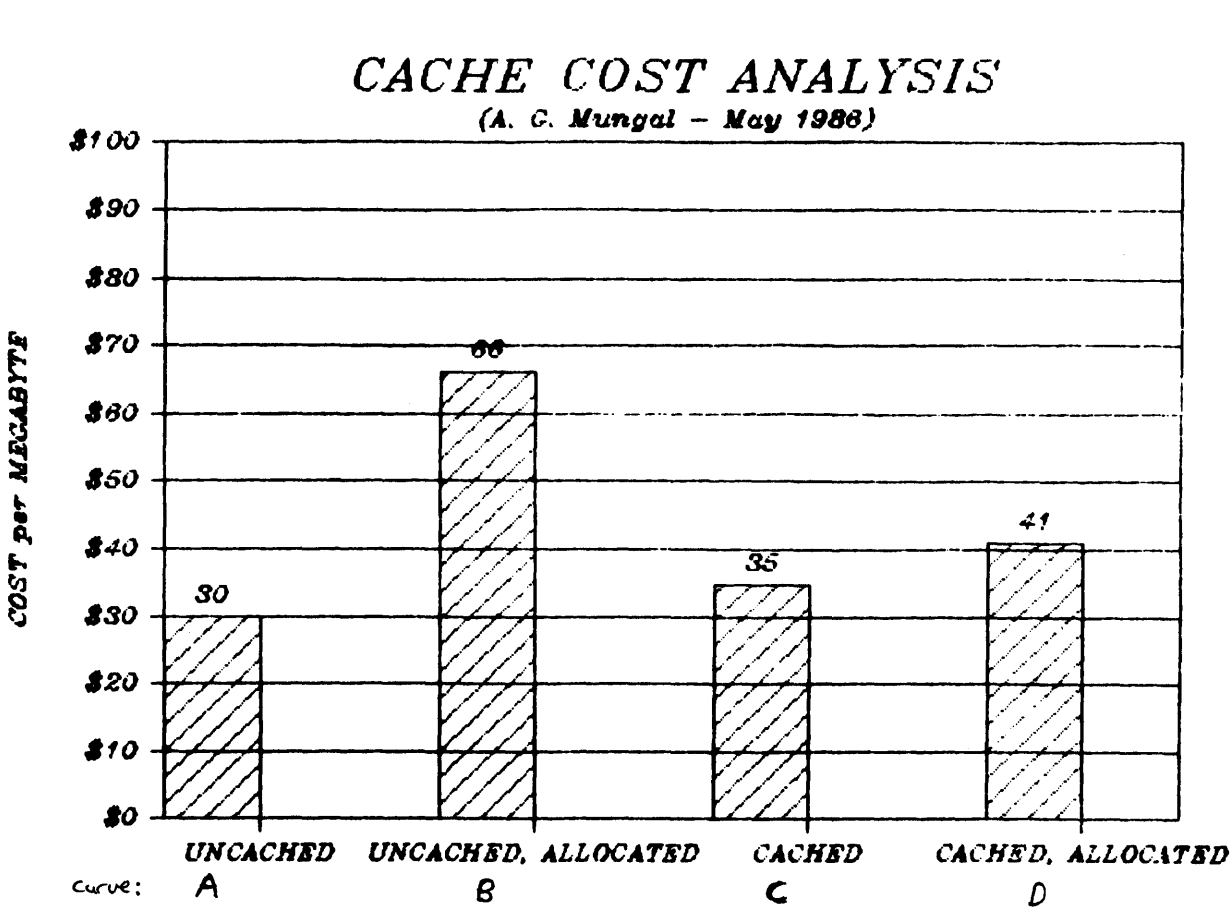


FIGURE 23 - COMPARING THE COST PER MEGABYTE.

and power; and also it provides the only real alternative to short strings.

APPLICABILITY OF SOLID STATE

Solid state subsystems share a subset of the high performance data configurable for caching. Data for solid state subsystems is characterized by high I/O rates, service time sensitive (low service time requirement), read to write ratio unsuitable for caching. The only concern for data on solid state subsystem lies in volatility; the issue of volatility is dealt with differently by various vendors of this technology.

The classic application area for the solid state disk has been, and continues to be, paging and swapping. This particular application area has been dealt with in other papers and is extremely well understood. Solid state subsystems, by virtue of their very low access times, the absence of seek and latency , have the ability to provide the best access density of all I/O systems being studied. It is this property that will allow for the most flexible and varied uses of this technology.

Even on processors configured with expanded storage, this technology can be used as a spill device. By employing solid state technology, one can insure that the access time disparity between expanded storage and the I/O subsystem is reduced. Mungal [11] suggests a method for utilizing solid state swap/page subsystems as a means of "effectively utilizing" main memory.

FUTURE CONSIDERATIONS

As high performance data continues to increase dragging the high reliability and availibility criteria, one must expect that devices employed in this arena will possess inherently high reliability and availability qualities. The issue of connectivity will also be of paramount importance in such subsystems. In coming to grasps with the requirements for low access times, one may expect that caches will become bigger, controllers smarter, and solid state subsystems to offer increased capacity and a myriad of pathing options. Disk subsystems may be expected to utilize smaller disks, provide buffering at the actuator level, and facilitate multi-porting (quad or octal).

Access density, which is the only true measure of I/O load and data set size, remains a useful measure for estimating the requirements of high performance subsystems. Utilizing access density as a means of evaluating the effectiveness of I/O subsystems, will provide the user with valuable information as to future configurations.

ACKNOWLEDGEMENT

I would like to thank my wife, Saroj for her painstaking efforts in the typing of this manuscript; also, for her patience and understanding during her many lonely hours while I prepared it.

REFERENCES

[1] McNutt, B. The Impact of 2-to-1 Volume Folding on Seek Time: A Comparison of Five Cases. In Proceedings of 1985 CMG International Conference (Dec. Dallas, Texas). 1985, pp.427-433.

[2] Thadani, A.J. Interactive User Productivity. IBM Systems Journal 20, 4 (1981).

[3] Moore, F. Solid State Device Considerations. In Proceedings of the 1985 CMG International Conference (Dec. Dallas, Texas). 1985, pp.136-153.

[4] Moore, F. Assisting DASD Cache Buffer Analysis. Storage Technology Corporation, Louisville, Colorado. 1986.

[5] Moore, F. Report on the Large Systems Storage Hierarchy. Storage Technology Corporation, Louisville, Colorado. 1987.

[6] Mungal, A.G. Cached I/O Subsystems: Analysis and Performance. In Proceedings of the 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.42-62.

[7] Duhl, B. Notes on Performance Considerations for Single and Dual Density DASD. In Proceedings of the 1986 CMG International Conference (Dec. Las Vegas, Nevada). 1986, pp.710-715.

[8] Singh, Y., King, G.M., Anderson, J.W. IBM 3090 Performance: A balanced system approach. IBM Systems Journal 25,1 (1986).

[9] Gray, B. The Use of Analytic Modeling to Compare Various I/O Configurations. In Proceedings of the 1985 CMG International Conference (Dec. Dallas, Texas). 1985, pp.116-122.

[10] Friedman, M. Performance and Tuning of Cached I/O Subsystems. In Proceedings of the 1985 CMG International Conference (Dec. Dallas, Texas). 1985, pp.717-727.

[11] Mungal, A.G. On the Optimization of Memory in Large Interactive Environments. In Proceedings of the 1987 CMG International Conference (Dec. Orlando, Florida). 1987

[12] Duhl, B. In Pursuit of Dasd Data Path Alternatives. In Proceedings of the 1987 CMG International Conference (Dec. Orlando, Florida). 1987.

AUTHOR/SPEAKER

Anthony Mungal is currently employed within Systems Engineering with Amdahl Canada Ltd. Prior to that he spent six years with Storage Technology (StorageTek Canada Inc.) in various positions within Systems Engineering, including product specialist, SE management and Consulting SE. His background is in systems programming with emphases on performance and tuning, capacity planning, storage management, programming, and I/O subsystems. He is a graduate of the University of Toronto (majors in mathematics and computer science) and an active member of the Association for Computing Machinery (ACM) and CMG Ontario.

PAGING, SWAPPING AND EXPANDED STORAGE

Jeffrey P. Buzen
J. William Mullen
Annie W. Shum

ABSTRACT

The introduction of expanded storage in IBM 3090 systems has been accompanied by a number of changes to the MVS algorithms that manage paging and swapping. This paper discusses the rationale for these changes and then explains their impact on certain measurement data reported by RMF. In addition, the paper illustrates several important aspects of page and swap performance through use of an analytic model.

INTRODUCTION

MVS has always supported a variety of devices for paging and swapping. When expanded storage became available, it could have simply been treated as a new type of paging or swapping device. The new device would, of course, rank higher than all others in terms of speed, and it would clearly require its own device driver.

Once these changes were made, MVS could have operated as usual. That is, system programmers could have allocated page and swap data sets directly to the new "expanded storage device", the existing page and swap algorithms could have accessed these data sets without modification, and the reports produced by RMF on page and swap activity could have included measurements of the additional traffic to the new page and swap data sets.

Although the VM/SP operating system has, in essence, adopted this approach, MVS has not. Instead, MVS has chosen to treat expanded storage as an entirely new type of resource, requiring special algorithms for management and new formats for performance reporting. In addition, the algorithms that control paging and swapping in MVS have been modified to take advantage of certain special properties of expanded storage.

There appear to be at least four reasons why MVS and VM have followed different paths:

1. VM already provided a page migration mechanism, whereas MVS did not.

2. Given that MVS had to be modified to support page migration, other algorithm changes needed to exploit the special properties of expanded storage were presumably easier to justify.

3. MVS had a greater need to provide selective control of expanded storage for individual workloads such as IMS, TSO and batch.

4. Given that all the above changes were being made, there was the additional opportunity to install changes needed to support DB2, VSAM linear and Data in Virtual (DIV).

The next few sections of this paper discuss these points in greater detail.

PAGE MIGRATION

Both MVS and VM are designed to favor higher speed paging devices over devices with lower speeds. In MVS, higher speed devices are used first, and lower speed devices are only used when no other space is available.

VM goes even further. If high speed paging devices become full, pages that have resided on these devices for a long time can be migrated (i.e., moved) to lower speed devices to make room for new pages being removed from central storage.

Note that migration is a costly process since it requires one read and one write operation for each page or block of pages that are migrated. Nevertheless, if there is a substantial difference in speed among the available paging devices, the cost of migration can be more than offset by having the most important (i.e., the least recently used) pages on the highest speed devices.

With the advent of expanded storage, the overhead of the migration process was substantially reduced (the estimated page transfer time between central and expanded storage is only 75-80 microseconds [2]). Also, the speed difference between expanded storage and conventional auxiliary storage could now be more than two orders of magnitude. These two factors made it essential to modify MVS to support page migration once expanded storage became available.

PRIMARY AND SECONDARY WORKING SETS

The designers of MVS have also introduced a number of other changes that can, in principle, substantially improve system performance by exploiting the special properties of expanded storage. One such change involves control of the swap-out/swap-in process. In systems without expanded storage, one of the critical steps in the swap out process is identifying the "working set" pages of the address space being swapped. These are the pages that the address space will need when it is swapped back into central storage and resumes execution.

To improve efficiency, working set pages are swapped out in contiguous groups to all available swap devices. When a swap-in is required, all working set pages are transferred back to central storage before the address space begins execution. In this way, address spaces can avoid generating costly demand page faults for pages that are part of their working sets.

Note, however, that demand page faults are not at all costly if the required page is in expanded storage. In fact, the CPU overhead to move a page from expanded to central storage is approximately the same for both page-in and swap-in operations. Hence, there is no real advantage to swapping the entire working set back into central storage before allowing an address space to resume execution.

MVS architects exploited this fact by introducing a distinction between primary and secondary working sets. The primary working set for a task contains only its LSQA pages and fixed pages. Secondary working sets are all pages owned by the task but not included in the primary working set.

At swap-out time, if the task is going to expanded storage (rules discussed later), the working set for the task is trimmed and the pages are transferred to expanded storage.

At swap-in time, only the primary working set for the task is swapped in. Thus, the task can resume execution more quickly, reducing response time.

After task swap in, the secondary working set pages remain in expanded storage and may subsequently be brought in via demand paging. In addition to improving response time, this algorithm also makes better use of central storage since only those pages actually used by the task are ever paged back in.

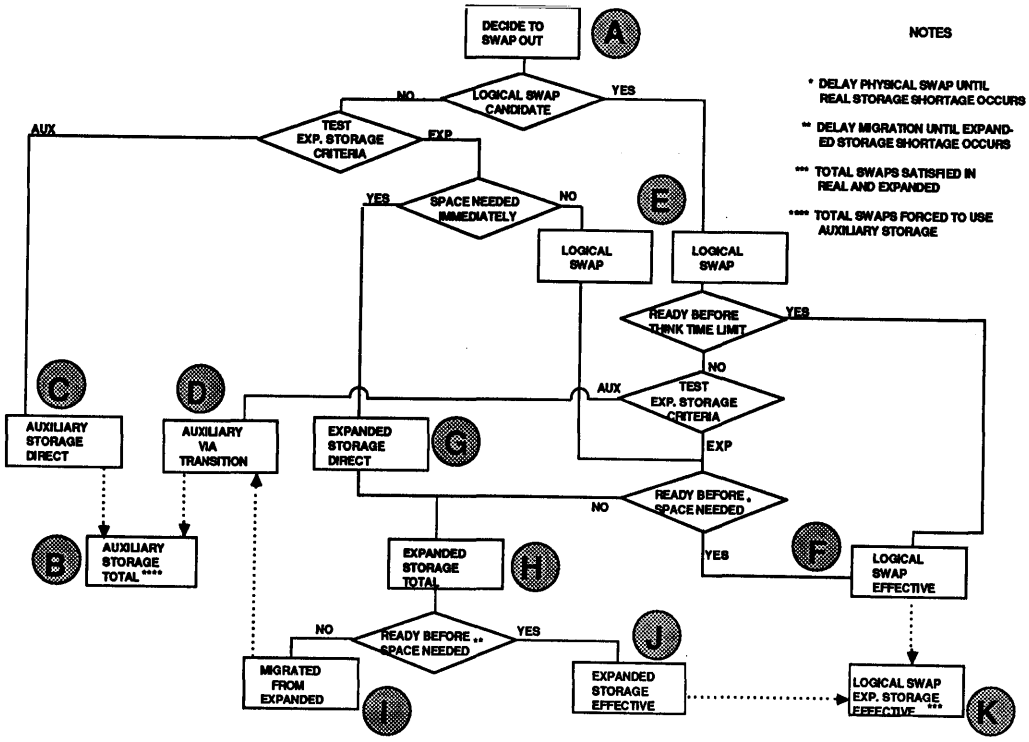


Figure 1

SWAP OUT ALGORITHM

A number of other changes have been made to the algorithms controlling logical and physical swapping in MVS. The flowchart in Figure 1 presents an overview of these algorithms. The flowchart begins at Box A when the decision is made to swap an address space out of central storage. The swap out decision could be the result of a variety of conditions including terminal input/output wait, long wait, detected wait, unilateral swap, exchange swap, etc.

Once the swap decision is made, the next step is to determine whether or not the address space is a logical swap candidate. For interactive tasks, this depends on whether the user's think time is less than the value of system think time.

Suppose that the address space is a logical swap candidate. In this case, the right branch (yes) below Box A is taken and the address space proceeds to Box E, Logical Swap. The address space is then allowed to remain in central storage until the "think time limit" expires. For interactive tasks, this limit is equal to the current value of the system think time. For batch tasks, this limit is a fixed constant (MVS currently uses 5 seconds).

If the address space becomes ready before the think time limit is reached, there is no need to carry out a physical swap. Instead, the address space proceeds to Box F (Logical Swap Effective) and is re-dispatched.

Thus far the existence of expanded storage has not affected any aspect of the basic algorithm. Suppose, now, that the address space does not become ready before the think time limit expires. In this case, a test of the expanded storage criteria is made to determine if the address space should be swapped to auxiliary or expanded storage. The nature of this test will be discussed later.

Suppose that the test indicates that the swap is to auxiliary storage. The address space will then proceed to Box D (Auxiliary via Transition) and be swapped out. The term "transition" refers to the fact that the address space has made a transition from a logical to a physical swap.

On the other hand, suppose that the test indicates that the swap is to expanded storage. In this case the original swap out algorithm is modified slightly. Note that expanded storage functions as an exceptionally fast swapping device; since the address space can be swapped out to expanded storage so rapidly, there is no need to initiate the swap out immediately unless central storage is needed by some other task. Hence, if central storage space is not needed, the address space is given a "second chance" and allowed to remain logically swapped even though the think time limit has expired. This "second chance" can be quite valuable since it allows systems to react immediately to fluctuations in the demand for real storage.

If, after its second chance, the address space becomes ready before the space it occupies in central storage is needed, there is an opportunity to proceed to Box F (Logical Swap Effective) as illustrated in Figure 1. On the other hand, the address space's page frames in central storage may be needed before the ready condition occurs, and there may, in fact, be a swap out to expanded storage via Box H, Expanded Storage Total.

At this point it is helpful to break the flow and return to the decision point below Box A. Now consider address spaces that are not logical swap candidates. Taking the left branch, the first test determines whether the address space should be swapped directly to auxiliary storage (Box C) or to expanded storage. Following the right hand branch at this point implies that the address space should be swapped to expanded storage. Once again, the original swap algorithm is now modified since there is no need to swap the address space out immediately unless the page frames it occupies in central storage are needed for other purposes. If the space is needed, the address space is swapped directly to expanded storage (Box G) and counted in the expanded storage total (Box H).

On the other hand, suppose the space in central storage is not needed immediately. In this case, the algorithm gives the address space a "second chance" and allows it to remain in central storage. However, the only means available for providing this second chance is to mark the address space as "logically swapped" (second copy of Box E). Thus, address spaces that are not logical swap candidates can, in fact, be logically swapped under this algorithm.

Once an address space undergoes this special logical swap, it is treated the same way as an address space given a second chance after exceeding its think time limit. That is, it either undergoes a swap to expanded storage (Box H) when its space is needed, or it completes a successful logical swap (Box F) if it becomes ready before its space is needed.

PAGING ACTIVITY

OS/VS2 SP2.1.3 SYSTEM ID RPT VERSION 3.3.0 START 02/18/86-10.14.01 INTERVAL 000.59.59 END 02/18/86-10.59.01 CYCLE 1.000 SECONDS

OPT = IEAOPT00

SWAP PLACEMENT ACTIVITY

		TOTAL (A)	AUX STOR TOTAL (B)	AUX STOR DIRECT (C)	AUX STOR VIA TRANSITION (D)	LOG SWAP (E)	LOG SWAP EFFECTIVE (F)	EXT STOR DIRECT (G)	EXT STOR TOTAL (H)	MIGRATED FROM EXT STOR (I)	EXT STOR EFFECTIVE (J)	LOG SWAP /EXT STOR EFFECTIVE (K)
		------ AUX STORAGE -------				*---LOGICAL SWAP--*		*---------- EXTENDED STORAGE ----------*				
TERMINAL	CT	55,473	955	302	653	51,879	37,660	3,292	17,246	388	16,858	54,518
INPUT/OUTPUT	RT	15.41	0.27	0.08	0.18	14.41	10.46	0.91	4.79	0.11	4.68	15.14
WAIT	%	91.4%	1.7%	31.6%	68.4%	93.5%	72.6%	5.9%	31.1%	2.2%	97.8%	98.3%
LONG	CT	1,352	31	3	28	1,230	845	119	501	25	476	1,321
WAIT	RT	0.38	0.01	0.00	0.01	0.34	0.23	0.03	0.14	0.01	0.13	0.37
	%	2.2%	2.3%	9.7%	90.3%	91.0%	68.7%	8.8%	37.1%	5.0%	95.0%	97.7%
DETECTED	CT	2,746	106	48	58	2,580	1,560	118	1,116	36	1,080	2,640
WAIT	RT	0.76	0.03	0.01	0.02	0.72	0.43	0.03	0.31	0.01	0.30	0.73
	%	4.5%	3.9%	45.3%	54.7%	94.0%	60.5%	4.3%	40.6%	3.2%	96.8%	96.1%
UNILATERAL	CT	1,038	45	29	16	928	548	81	445	0	445	993
	RT	0.29	0.01	0.01	0.00	0.26	0.15	0.02	0.12	0.00	0.12	0.28
	%	1.7%	4.3%	64.4%	35.6%	89.4%	59.1%	7.8%	42.9%	0.0%	100.0%	95.7%
EXCHANGE ON	CT	27	0	0	0	26	9	1	18	0	18	27
RECOMMENDA-	RT	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.01	0.01
TION VALUE	%	0.0%	0.0%	0.0%	0.0%	96.3%	34.6%	3.7%	66.7%	0.0%	100.0%	100.0%
ENQUEUE	CT	45	2	2	0	38	23	5	20	0	20	43
EXCHANGE	RT	0.01	0.00	0.00	0.00	0.01	0.01	0.00	0.01	0.00	0.01	0.01
	%	0.1%	4.4%	100.0%	0.0%	84.4%	60.5%	11.1%	44.4%	0.0%	100.0%	95.6%
TRANSITION	CT	3	3	3	0	0	0	0	0	0	0	0
TO NON-	RT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SWAPPABLE	%	0.0%	100.0%	100.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
TOTAL	CT	60,684	1,142	387	755	56,681	40,645	3,616	19,346	449	18,897	59,542
	RT	16.86	0.32	0.11	0.21	15.75	11.29	1.00	5.37	0.12	5.25	16.54
	%	100.0%	1.9%	33.9%	66.1%	93.4%	71.7%	6.0%	31.9%	2.3%	97.7%	98.1%

Figure 2

Now consider all address spaces that eventually become swapped to expanded storage (Box H). If they remain in expanded storage too long, the space they occupy may eventually be needed by other address spaces. In this case they are migrated to auxiliary storage (Box I). The dotted line connecting Box I to Box D indicates that migrated address spaces are also included in the total number reaching auxiliary storage via transition.

On the other hand, if an address space becomes ready before it is migrated, its primary working set is swapped back immediately to central storage (Box J). This box is labeled "Expanded Storage Effective" since it represents a desirable case where the swap-out/swap-in cycle is handled entirely by expanded storage.

Box K (Logical Swap/Exp. Storage Effective) represents the sum of Boxes F and J. In effect, it is a count of the total number of high performance swap cycles. In contrast, Box B (Auxiliary Storage Total) is the sum of Boxes C and D. It represents a count of all low performance swap cycles that used conventional auxiliary storage devices.

SWAP PLACEMENT ACTIVITY REPORT

The flowchart illustrated in Figure 1 bears a direct relationship to the RMF Swap Placement Activity Report shown in Figure 2. Each box in the flowchart corresponds to a column heading in the report, as indicated by the associated letters. The number of times that address spaces make transitions through the various boxes in the flowchart is represented both by raw counts and by percentages in this RMF report.

EXPANDED STORAGE CRITERIA

There are two decision points in this flowchart where the "Expanded Storage Criteria" are tested to determine if an address space should be swapped out to auxiliary or expanded storage. These tests are based on the IEAOPTxx parameter values in the Expanded Storage Criteria Table (ESCT). They also depend on the current values of the system high UIC, the migration age (MA), and the address space's think time.

In general, if there is space available in expanded storage, the address space will be swapped to expanded. If expanded storage is almost full, the address space may be swapped to auxiliary storage instead. See [1] for a further discussion of algorithms that operate on the ESCT.

MEASUREMENT OF PAGE/SWAP ACTIVITY

Paging and swapping activity in MVS is recorded by the SRM and subsequently retrieved by RMF for generation of the RMF type 71 record and the Paging Activity Reports. Introduction of the 3090 processors and expanded storage required a restructuring of the RMF type 71 record to provide usage statistics for expanded storage. Additional information relating to migration age and to page movement to expanded storage were added to the paging activity section of the record. Swapping activity for each of the 11 swap reason codes was removed from the body of the record and placed in the record's relocate sections. Each of the 11 relocate sections now contains swap placement information, including expanded storage usage, for the respective swap reason code.

The RMF Paging Activity Report was updated with Release 3.3 of RMF to reflect the new measurement values available in the RMF type 71 record. Two new values of interest to the performance analyst are the 'Rate of Page Movement to ES' and the 'Migration Rate'. Each of these values is reported as pages-per-second for the measurement interval. As a guideline, it has been suggested that the processor capacity used to transfer pages to and from expanded storage should not exceed a maximum of 5%. Using a value of 75-80 microseconds per page, this would imply that a 3090-400 processor should not have a transfer rate to and from expanded storage that exceeds 2500 pages per second ((1,000,000 * 4 processors * .05) / 80). This guideline must obviously be reconciled with other considerations such as the service objectives for the workloads being processed, and the impact that page movement has on these service objectives. In 303X and 308X systems without expanded storage, some performance analysts have traditionally recommended as a guideline that no more than 5% of processor utilization be devoted to demand paging operations. It is interesting to note that the same 5% value is now being recommended for 3090 systems with expanded storage.

A third value added to the Paging Activity Report is the minimum, maximum and average values of migration age for the measurement interval. These values are an indication of the contention for expanded storage frames. Average migration age also affects the probability of certain types of paging and swapping operations being directed to expanded or auxiliary storage, as explained previously.

The fourth value added to the Paging Activity Report is the number of installed and online expanded storage frames during the interval. If the online expanded storage frames reported for the interval is less than the installed expanded storage frames reported for the interval, the performance analyst should determine the cause.

The Swap Placement Report was replaced in RMF Release 3.3 with a new report identified as the Swap Placement Activity Report. The new report format, which is illustrated in Figure 2, was needed to report the various swap transition rates for each of the eligible swap reason codes and to provide the additional information on swapping activity to expanded storage. The Swap Placement Activity Report contains swapping information for only 7 of the 11 swap reasons since some of the swap conditions are not candidates for logical swapping or swap to expanded storage.

One statistic of interest to the performance analyst is 'LOG SWAP EFFECTIVE' (column F). This column contains the rate of logical swaps per second and the percentage of tasks logically swapped out that were subsequently swapped in from the logically swapped out state. A high percentage for this statistic indicates that contention for real storage is low.

A second important statistic is the'LOG SWAP/EXT STOR EFFECTIVE' (column K). A high percentage for this statistic indicates that contention for expanded storage is low and frames are plentiful. As this percentage decreases, the analyst will note a decrease in 'EXT STOR EFFECTIVE' (column J). The latter is most undesirable since it indicates that contention for expanded storage frames is growing and forcing the MVS/XA Real storage Manager (RSM) to remove pages from expanded storage (migration) and place them on auxiliary storage paging data sets.

Several statistics that are available to RMF for data capture and subsequent reporting are conspicuously missing from the Paging Activity Reports. The minimum, maximum and average system 'think time' for the measurement interval could be added to the Paging Activity Report to provide the analyst with a better view of the SRM decision making processes for the interval.

The RMF type 71 record contains a count of the pages written to expanded storage as well as a count of the pages migrated from expanded storage. These counts are converted to rates (pages/sec) and reported in the Paging Activity Report. It would be very desirable to add measurements of page-in rates from expanded storage to this report.

Information on individual task usage of expanded storage or specific types of task usage of expanded storage is currently available through the RMF type 79, subtype 1 (ASD) Monitcr II recording facility. This information could be accumulated in the RMF type 72 records for subsequent reporting in the RMF Workload Activity Report as minimum, maximum and average expanded storage frames utilized by performance group and domain during the interval. MVS/XA Release 2.2, available 3Q87, is supposed to provide additional information on expanded storage usage.

PAGE AND SWAP INTERACTIONS

Expanded storage adds one new dimension to the already complex process of characterizing paging and swapping behavior in MVS. Some of the major factors that affect this behavior will now be examined with the aid of a queueing network model. The model incorporates facilities for automatically evaluating paging and swapping behavior as various parameters are altered.

Consider first an MVS system without expanded storage. Suppose the external load on the system is held fixed while the size of central storage increases. As expected, paging and swapping are reduced in this case. Figure 3 illustrates this reduction for a typical model.

Even in this familiar case, note that two distinct factors are operating. First, demand paging is reduced since each active address space is able to acquire more page frames. This increase in resident set size implies that each address space will encounter less page stealing. Thus, the number of page faults caused by references to stolen pages is reduced. Figure 4 illustrates the nature of this interaction.

Swapping is also reduced in this case, but for a different reason. Essentially, increasing the size of central storage increases the system think time, allowing logically swapped address spaces to postpone the need for a physical swap. If the address space becomes ready soon enough, a physical swap is avoided. Figure 5 illustrates the nature of this process.

Note that other factors are also operating. For example, since the demand paging rate is reduced, each address space requires less time in central storage per transaction. Thus, the average multiprogramming level is reduced as illustrated in Figure 6. This in turn reduces the demand for real storage, allowing resident set sizes to increase further and logical swapping to be even more effective. The queueing model used in this example automatically accounts for these factors and incorporates them into the graphs that appear in Figures 3, 4 and 5.

A similar interaction between multiprogramming level, paging rate and swapping rate would arise if CPU or I/O device speeds were increased. This increase would, in general, reduce residency time. This would, in turn, reduce multiprogramming levels, making additional space available for active resident sets and for logical swapping. This would then reduce paging and swapping rates. The automatic analysis of these interactions is of real value in capacity planning applications.

Now consider a different set of interactions that occur when storage size is held fixed while the transaction arrival rate for interactive applications increases. As Figure 7 illustrates, both paging and swapping rates increase in this case.

Paging increases because, as the arrival rate increases, there is more queueing for the CPU and the I/O devices in the system. Thus, each transaction requires more time in central storage (i.e., residency time per transaction increases). This in turn increases the average multiprogramming level. Since there are more active address spaces at each instant, MVS will have to increase page stealing to reduce resident set sizes. This will cause more demand page-in's for stolen pages as Figure 7 illustrates.

Swapping will also increase in this case since the increase in transaction arrival rate is typically accompanied by an increase in the number of interactive users. With more logically swapped out users competing for space in central storage, MVS will reduce system think time and force physical swaps to occur sooner. This interaction is also accounted for in the queueing model used to generate Figure 7.

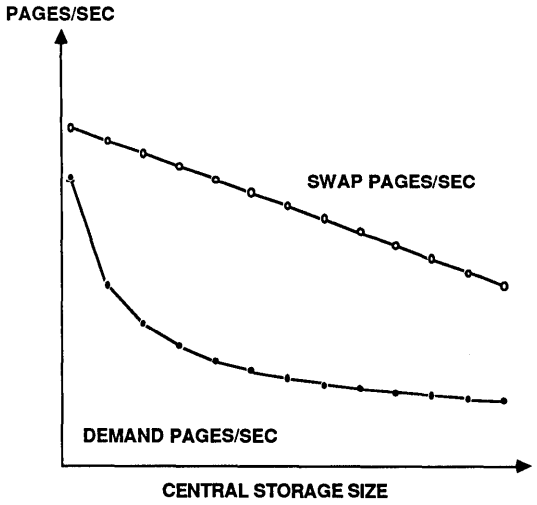


Figure 3

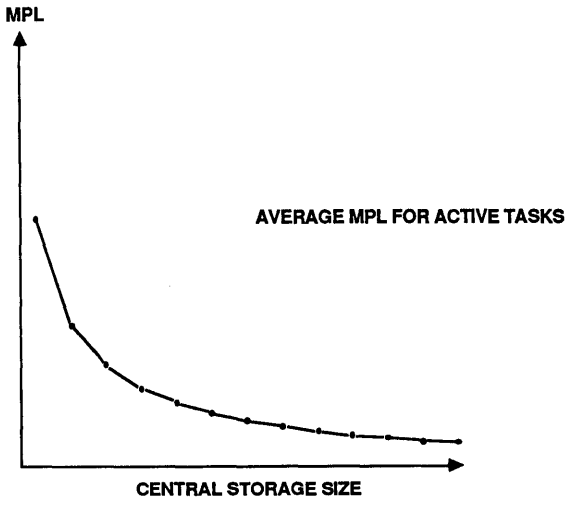


Figure 6

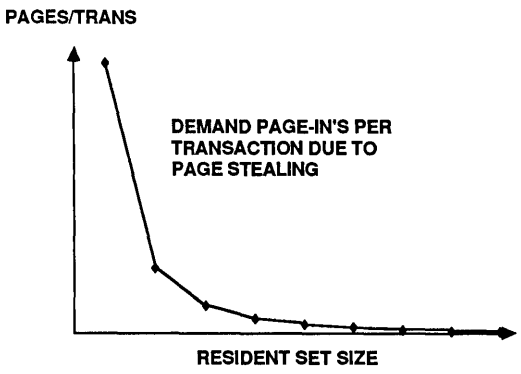


Figure 4

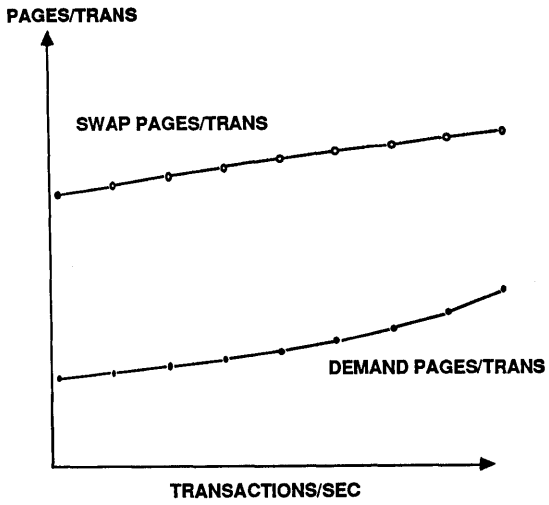


Figure 7

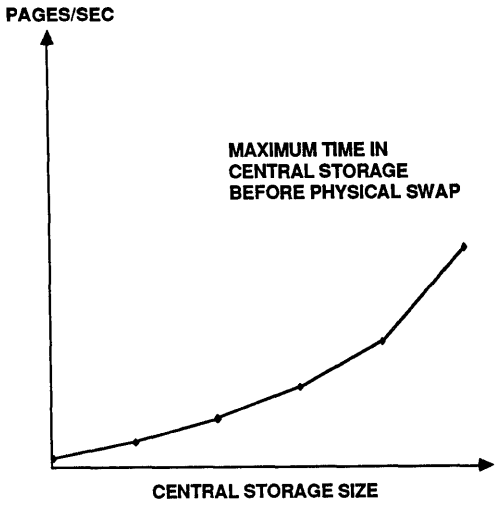


Figure 5

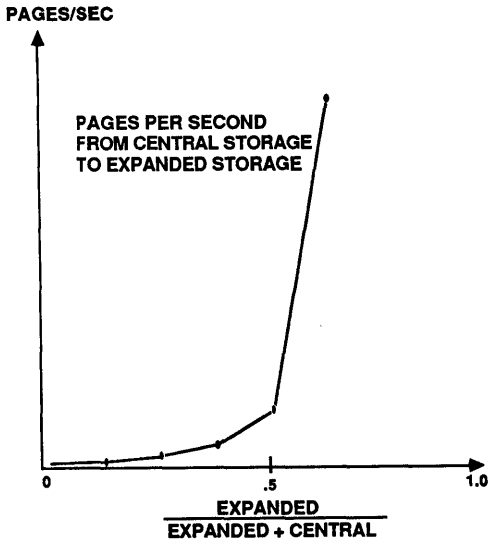


Figure 8

IMPLICATIONS OF EXPANDED STORAGE

Now consider the effect of expanded storage on the interactions illustrated in Figures 3-7. Note first that systems with expanded storage can be analyzed at two levels:

Level 1 - Consider only the total amount of processor storage (central storage plus expanded storage), and focus only on page and swap traffic between processor storage and conventional auxiliary storage devices.

Level 2 - Provide additional analysis of page and swap traffic between central storage and expanded storage, and also analyze the page migration process.

In a Level 1 analysis, the overhead required to transfer pages between central and expanded storage is ignored. Thus, the major concern is page and swap traffic to auxiliary storage.

As has been noted earlier and illustrated in Figure 1, pages may be sent to auxiliary storage directly from central storage, or they may be sent indirectly from expanded storage because of migration.

Essentially, migration occurs if a page is being removed from central storage at a time when expanded storage is "nearly full." Note, however, that direct transfers from central to auxiliary storage also occur if a page is being removed from central storage at a time when expanded storage is "nearly full." The only difference is that, in the direct case, the page itself is transferred to auxiliary storage; in the case of migration, a different page is transferred.

Since, in either case, page transfers to auxiliary storage only occur when both central and expanded storage are nearly full, a simple Level 1 analysis need only be concerned with the total amount of central plus expanded storage (i.e., the total amount of processor storage). If the term "central storage" is replaced by the term "processor storage" in the discussion of Figures 3-7, then all the observations and conclusions will apply equally well to a simple Level 1 analysis of a system with expanded storage.

The queueing model used to generate Figures 3-7 does, however, have to be enhanced to carry out a Level 2 analysis. One required extension is predicting the amount of page traffic between central and expanded storage. In general, if the total amount of processor storage is held constant, this page traffic will increase as the relative proportion of expanded storage increases. Figure 8 illustrates this effect. [Note, of course, that a system can never contain 100% expanded storage. MVS/XA needs a substantial amount of real storage to operate effectively.]

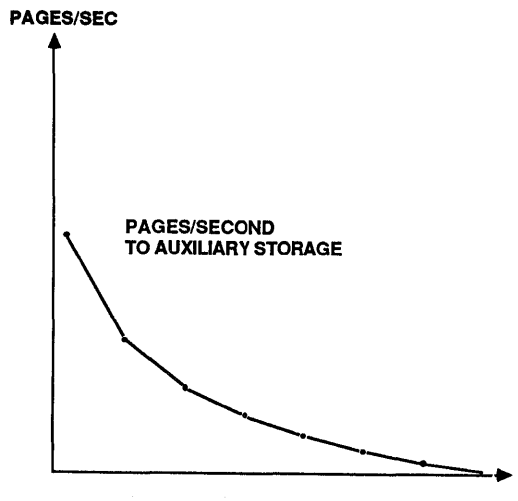


Figure 9

Figures 9-12 illustrate another interesting interaction involving real and expanded storage. The figures represent a case where the size of real storage is held constant while the size of expanded storage is steadily increased.

As illustrated in Figure 9, increasing the size of expanded storage causes a decrease in the number of demand pages per second that go to auxiliary storage. This result is entirely expected.

One might also expect that the reduction in demand pages per second to auxiliary storage would be accompanied by a corresponding increase in the rate of demand pages per second to expanded storage. Figure 10 shows this relationship.

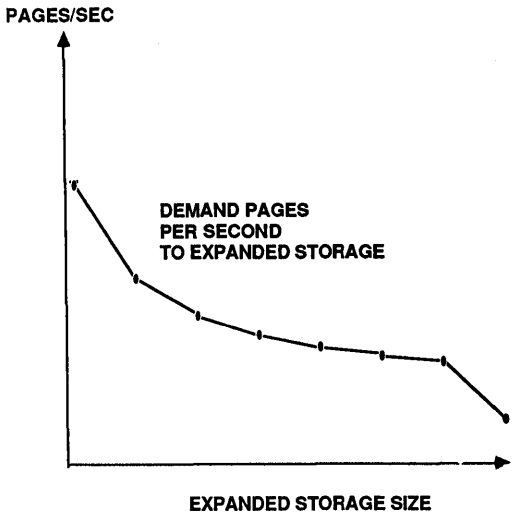


Figure 10

Surprisingly, the rate of demand pages per second to expanded storage does not increase: instead, it shows a distinct decrease, with a sharp acceleration at the point where all demand paging is handled by expanded storage.

To understand this phenomenon, note first that expanded storage is obviously much faster than conventional auxiliary storage. As the amount of expanded storage increases, it is able to handle a higher and higher percentage of total demand paging. Thus, the response time per transaction will steadily decrease.

As response time decreases, the amount of time each transaction occupies real storage will, of course, decrease. Thus, the average multiprogramming level (i.e., the average number of active transactions in real storage) will also decrease. This effect is illustrated in Figure 11.

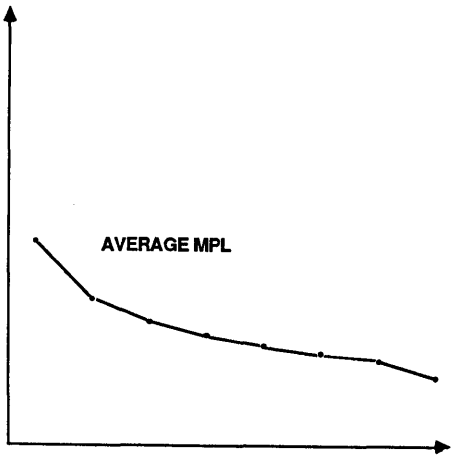


Figure 11

Finally, decreasing the average multiprogramming level increases the amount of real storage available to each active transaction (since there are fewer active transactions at any given instant). Thus, the average resident set size per transaction will increase. Figure 12 illustrates this increase.

Figure 12

Since resident set size increases, there is less page stealing. Thus, the total number of demand page-in's per transactions decreases. This is the key to explaining the decreasing paging rate in Figure 10.

The total demand paging rate is obtained by adding together the auxiliary storage paging rate in Figure 9 and the expanded storage paging rate in Figure 10. Figure 13 presents this total rate.

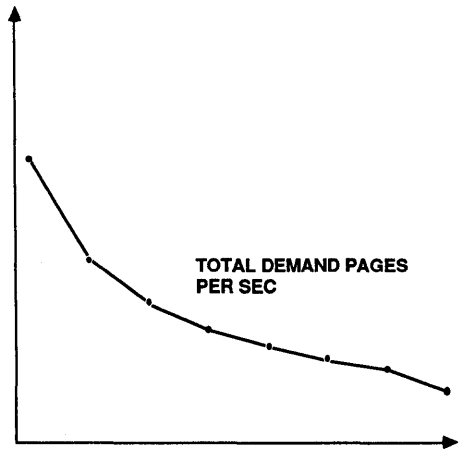


Figure 13

As expected, the total rate declines as expanded storage size increases. This, of course, follows from Figure 12 since resident set size increases steadily. Moreover, the decreasing rate in Figure 13 is more than enough to offset the increasing proportion of demand paging that is being handled by expanded storage. Hence, Figure 10 has a very reasonable explanation even though it may first appear to be counter-intuitive.

As the example demonstrates, the use of powerful analytic models can be invaluable in analyzing the complex interactions that occur in high performance computer systems. The ability to carry out these analyses in an accurate and fully automatic manner can be of substantial benefit to anyone working in the area of capacity planning or performance management.

SUMMARY

A number of changes have been made in MVS to accommodate expanded storage. These changes have been discussed, and their impact on the RMF Swap Placement Activity Report has been reviewed in detail. Also, an analytic queueing model has been used to illustrate important aspects of paging and swapping behavior that arise in systems with, and without, expanded storage.

REFERENCES

[1] Buzen, J.P., Mullen, J.W. and Shum, A.W., "Measurement and Analysis of Expanded Storage in IBM 3090-400 systems." CMG 87 Proceedings, this volume, see Table of Contents for page numbers.

[2] Tucker, S.G., "The IBM 3090 System: An Overview," IBM Systems J., 25, 1, 1986, pp. 4-19.

PERFORMANCE EVALUATION OF FUTURE SYSTEM DESIGNS

J.M. Quernemoen, M.A. Fahland, B.G. LaPlante and T.L. Nelson
UNISYS Corporation

ABSTRACT

This paper presents a method of evaluating the performance of future computer systems. Although the design and development of a computer is a process that generally is not visible to customers, much of the methodology is relevant to the performance evaluation of existing systems. Presented is an established means of determining computer performance of systems that exist only as design concepts.

The paper covers three topics:

1) Workload Selection, Characterization and Measurement
2) Automated Discrete Event and Analytic Modeling Tools
3) A Hierarchical Approach to Modeling Future Systems from the Machine Cycle to End-user Response Time

1 INTRODUCTION

Performance prediction has become an integral and necessary part of the computer design process. The performance evaluation of increasingly complex systems and their components, as they evolve from design concepts into prototypes and products, provides a dynamic means to foresee and prevent performance-related design shortcomings. To successfully predict the performance of future systems, one must use meaningful performance metrics, and the correct tools and methodology.

Measuring and predicting performance have much in common. They both provide an evaluation of how well a system operates and why it performs as it does. They both evaluate the system with the same set of performance metrics and with the same sets of workloads, both of which are used to specify performance requirements before design begins, to estimate performance during the design, and to verify performance after the product is built.

These workloads must be representative of real applications in order to be meaningful. They must also be repeatable. To satisfy these requirements appropriate workloads or sets of workloads are developed into standard system benchmarks which are used as the basis for system performance measurement and prediction.

Performance measurement is accomplished by running benchmarks on existing systems. Performance prediction consists of running a model of the system driven by the workload characteristics of these benchmarks. Therefore, part of the modeling process is to decide what characteristics of the workloads are required as input to the model and to measure those characteristics on an existing system. Section 2 discusses workload selection, characterization, and measurement.

Finally come tasks of developing a complete and concise model definition, choosing the proper modeling technique, and building and running the model. Section 3 describes automated modeling tools that were developed to facilitate and expedite these tasks.

For each component of the system being modeled, one must evaluate the level of detail required to adequately represent the performance-related aspects of that component. Certain components may need to be simulated at the nanosecond level, others at the millisecond level. Mixing these levels in a single model poses serious modeling problems, yet ignoring important details could result in undetected performance problems. To avoid mixing widely different levels of detail in a single model, we use a methodology that uses results from detailed models as input to higher level models. This hierarchical modeling methodology is described in Section 4.

2 WORKLOAD SELECTION, CHARACTERIZATION, AND MEASUREMENT

Workloads representative of those that are expected to run on the new system are used throughout the system design process. They are first used in specifying the performance requirements for the new system. Then the workload characteristics are determined, and used as input for models. Finally, the workloads are run on the new system to verify that the performance goals have been met. This section discusses how we select, characterize, and measure workloads.

2.1 SELECTION CRITERIA

Workloads are selected to do one or more of the following:

- o Represent actual customer environments
- o Stress some aspect(s) of the system architecture
- o Gauge the performance of system software

The first of these is the most important. To sell a system, you must be able to tell the customer how it will perform in his environment. Workloads may be selected to represent banking, airlines, business, or scientific environments, depending on how the system will be marketed. These workloads come from existing customers, or are based on measurements made at customer sites.

When there is a known potential bottleneck in the system, or when doing trade-off studies on a particular component, a workload may be selected which stresses that part of the system. For example, in evaluating the main storage system, we would use a workload with high memory bandpass requirements.

Lastly, workloads must be available to assess the performance of system software, including the operating system, compilers, linkers, and applications such as communications, database management, and transaction processing. For instance, standard workloads are used to evaluate every compiler release, based on the efficiency of both the compilers and the oode generated.

Of course, today's workloads do not always represent what we expect to be running in the not-too-distant future. Workload characteristics are changing due to the following factors: off-loading of certain tasks onto personal computers or specialized processors; increasing memory size, thus allowing higher concurrency levels, larger programs, and all kinds of caching; and increasing peripheral performance, thereby raising memory bandpass requirements. Thus, the design of tomorrow's systems is based on today's workload characteristics with some over-design and extrapolation for the expected characteristics of tomorrow's workloads.

2.2 WORKLOAD CHARACTERIZATION

In order to use models to predict how well a system or its components will operate under a given workload, the performance-related characteristics of the workload must be understood. Three areas commonly profiled are IP characteristics, I/O characteristics, and memory characteristics. The detail of the characterization is directly related to the detail of the model.

The IP profile of a workload includes the sequence of instructions executed, the frequency of various instructions or classes of instructions (such as load, store, jump, arithmetic, string, and memory-to-memory), and the number and type of operands. This information provides the basis for predicting IP and memory organization performance.

I/O profiles include the frequency of mass storage I/Os, pathlengths between I/Os, transfer sizes, read/write ratios, and load distributions across physical devices. This information is used to determine the design requirements for a system's I/O complex: bandpass and throughput, number of channels, number of I/O processors (IOPs) and performance requirements of the IOPs, disks and disk controllers.

Memory profiles include the frequency of memory references, the distribution of reference type (read, write, lock, etc.), and locality of references. These characteristics of a workload are used in evaluating performance of memory hierarchies, memory interleaving, and paging algorithms.

2.3 WORKLOAD MEASUREMENT

Workload characteristics required as input by performance models are generally derived from measurements performed on existing systems. This section covers three ways of measuring workloads.

2.3.1 SOFTWARE MONITORING. The Unisys Software Instrumentation Package (SIP) provides system-level performance measurements for 1100 Series products. SIP is a built-in software monitor which can be turned on and off at the system console. SIP writes the data collected to an output file at the end of regular time intervals. The length of the interval is selected by the operator in granules of seconds. While SIP collects many types of information, much of it is geared toward system tuning and capacity planning. Our description of SIP's capabilities will be limited to the I/O statistics it collects.

SIP collects data on I/O activity by channel, control unit, and device. Data collected include software queueing times, hardware service times, data transfer times, input and output request counts, and number of words transferred.

A feature of SIP called IOTRACE provides a record of each I/O activity occurring while it is active. IOTRACE can run independently or with other SIP data collection. In the latter case, the IOTRACE data can be synchronized with the other SIP data. The trace information for each I/O operation includes:

- o Request initiation time
- o Requestor (program or operating system entity)
- o File/device identifier
- o File/device starting address
- o Request type (e.g., read or write)
- o Transfer size
- o Timing: software queueing time, seek time, hardware response time

This data, together with synchronous hardware monitoring of the system's IP utilization and instruction count, provides a relatively system independent picture of the candidate workload's I/O profile. Specifically, one can reduce this information to obtain:

- o I/O arrival rate distribution
- o Duration of selected I/O arrival rate ranges
- o Distribution of load on channels, control units, and devices
- o Mass storage utilization
- o Distribution of transfer sizes

This information, which can be obtained for any time interval within the sample period, is used as input to subsystem models, I/O complex models, and system models.

2.3.2 HARDWARE MONITORING. Although software instrumentation provides system-level performance metrics, it has two distinct shortcomings. First, it is unable to measure lower level metrics such as IP cache hit rates and instruction mixes. Secondly, it can add significantly to system overhead.

Hardware monitoring is used to obtain detailed measurements of such things as cache performance, instruction mixes, memory references, software pathlengths, and processor activity breakdown. Since hardware monitors are generally passive devices, they do not alter the operation of the system under test.

Hardware monitoring typically involves connecting probes to monitor locations identified by the manufacturer. Depending upon the type of measurement, a few signals or several dozen may be required. The signals are brought to the monitor equipment where they are used individually or in combination to control counters, timers, data storage, and real-time displays.

Four basic monitoring techniques are counting, timing, mapping, and storing. Most monitoring equipment provides a means of doing some combination of these. Comparators are often used in conjunction with all types of measurements to restrict the measurement to specific intervals.

Use of counters is the most common mode of hardware measurement. In counter mode, one or more signals are combined by some programmable logic in the monitor, and the resulting signal is used to increment a counter. Periodically, the counter values are written to disk or tape, and the counters are reset. Examples of performance data collected using counters are shown in Table 2.1.

When more counters are needed than are available in a monitor, a mapping technique is used. For example, to count the number of occurrences of each instruction type on a computer with an 8-bit function code would require 256 counters. Instead of using normal hardware counters, the binary value of the combined signals (0 to 255 in our example) is used as an offset into an area of memory. The value in the memory location at

Table 2.1. Examples of Counter measurements

Metric	Counters Required
Instruction Execution Rate	Instructions, Duration of Measurement
Cache Hit Rate	Cache References, Cache Hits
Instructions per I/O	Instruction Count, I/O Count
References per Instruction	Instructions, Memory References
Memory Reference Patterns	Memory References for each MSU or Bank
Pathlengths	Instructions

that offset is incremented. This technique is also used to determine frequently executed portions of code by monitoring memory address bits.

Hardware monitor timers are useful for measuring very short time durations such as memory access times, or the time from a request to an acknowledge. Typically, one signal is used to start the timer, another to stop it.

The store mode technique is used to gather details of consecutive events or instructions, such as instruction or memory address traces. In store mode, many signals are captured and stored on each event occurrence. Since mass storage is usually not fast enough to store data at a high rate, memory is generally used for holding these traces. When the available memory is full, it is written to disk or tape. The system being monitored usually must be suspended during the writing process, or events will be missed during the measurement.

2.3.3 HARDWARE SIMULATION. In addition to commercial hardware monitoring equipment, special purpose hardware is also used to collect highly-specialized types of data. One example of such hardware is the Unisys Buffer Information Collector (BIC).

BIC was designed in order to accurately predict the performance of future cache designs by simulating those designs in hardware. Cache simulation of large programs or benchmarks using software simulators was not feasible because of the large address traces required and the overhead of such simulations. Using BIC, cache performance for an entire benchmark can be evaluated in the time it takes to execute that benchmark.

BIC is a combination of hardware and micro-programmable components which simulate a wide variety of caches and provide several types of address traces. BIC operates by intercepting the memory address and function lines which connect an instruction processor and its local cache memory. The acknowledge from the cache is delayed by BIC until BIC's operations are complete. In normal simulation mode, the delay is minimal and has no impact on the program executing in the host.

BIC also contains a standard I/O channel interface which allows it to generate address traces as it is executing. The I/O channel is also used to download BIC microcode which is used to test the hardware, to verify that the cache algorithm is properly wired, and to aid in simulating caching algorithms not capable of being simulated in the hardware alone.

BIC has three modes of operation. Mode 1 is its standard simulation mode. The cache size and cache algorithms are selected via patch card jumpers. A hardware monitor connected to BIC records hits, misses, and reference counts for each of the caches being simulated. Mode 1 has been used to perform a large number of cache studies, including those shown in Table 2.2.

Table 2.2. Examples of BIC Mode 1 Measurements

Cache Organization
- Varying Cache Sizes
- Varying Block Sizes
- 16-word block with 8-word fetches

Caching Algorithms
- Store-in (Post-write)
- Store-thru (Write-thru)
- Prefetching
- Certain requests bypass the cache
- Various software locking algorithms

Aging Algorithms
- LRU
- FIFO
- Least Recently Used Pair

Multiple Caches
- Up to four caches
- Exec and User
- Instruction and Operand
- Buffer Invalidation

Modes 2 and 3 are used to collect address traces. Under preset conditions, the BIC hardware interrupts the BIC microprocessor, which then retrieves address and control signals and sends them out over the I/O channel. For example, by interrupting on only those references which would cause a reference to main store, BIC generates a complete trace of main store addresses and reference type (read, write, lock, etc.).

This address trace can then be used to evaluate second-level cache and memory designs. In addition, mode 2 provides a data compression technique, which can reduce the data and time required to evaluate paging and other higher-level memory management algorithms. BIC has also been used for special studies, such as simulating branch prediction buffers.

3 MODELING TOOLS

Computer performance modeling tools improve access to performance related data and automate the model building process. This section presents a selection of the tools we use: a database for managing the collection of performance data, a graphical model description language, and two automated model building packages.

3.1 PERFORMANCE DATABASES

Literally thousands of data collections have been made on the standard workloads we use for our performance predictions, as well as collections made at customer sites. To allow analysts to find what data has been collected and where the data resides, we have created a series of on-line databases (see Figure 3.1). They are implemented in MAPPER and are accessible nationwide via a proprietary development network.

The Collection Database describes each measurement in terms of what hardware and software were measured, and what data was collected. The Collection Database also has a 'Request for Measurement' function. If the information required does not exist, this function may be invoked to request that the data be collected. The request is recorded in the database, and its status is updated as it is scheduled and completed. When the measurement is complete, the measurement request record becomes the measurement description record and points to the measurement results.

The Collection Database also contains pointers to where the data is stored: in another database, in on-line files, or on magnetic tape. Much of the raw data from hardware monitor measurements is kept in on-line MAPPER databases. Having the data on-line allows the analyst to compute and compare the statistics needed for any number of cases. More voluminous data is not kept in the on-line MAPPER database; instead, standard reductions from hardware and software monitors are performed and kept in designated on-line files. All the raw data collected during a measurement is stored on magnetic tape.

Tools, designed to aid the analyst and modeler, have been integrated into the on-line databases using MAPPER's programming capability. The Instruction Trace Facility (ITF) is an application of the MAPPER programming capability. The ITF database contains detailed instruction trace information for each of the standard performance benchmarks, and instruction timings for each baseline machine and each machine currently under development. The tools associated with the ITF database include interactive runs which do the following:

- o Generate Instruction Mix Reports
- o Generate Instruction Sequence Reports
- o Analyze Operand Usage
- o Calculate base average instruction execution time for any combination of benchmark and machine in the database.
- o Generate customized instruction trace files to be used as input to discrete event models. The user defines the data needed and the appropriate format.

These database tools provide answers to "what if" questions as well. By supplying parameters such as timings, mixes, and instruction class definitions, the analyst can determine the effect of changes to hardware and software.

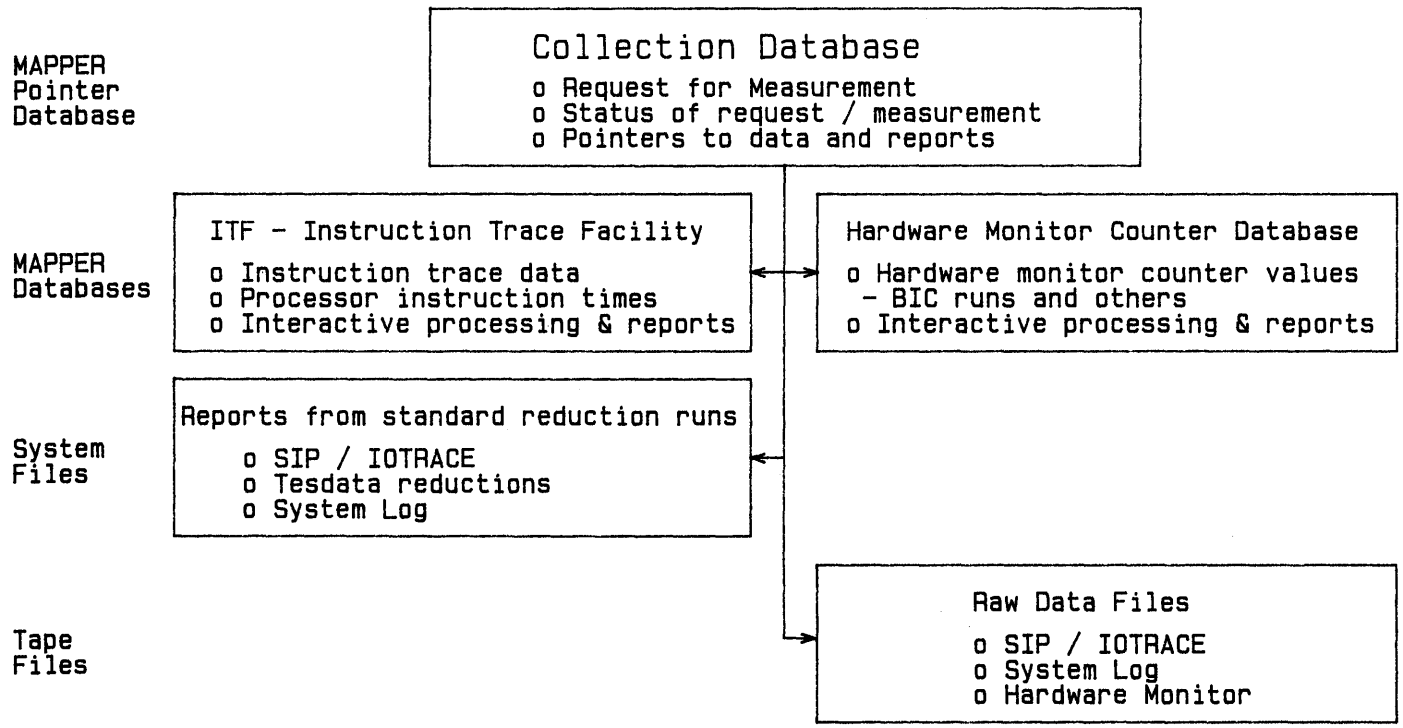


Figure 3.1. Performance Databases

3.2 AUTOMATED MODEL-BUILDING TOOLS

During the feasibility and design stages of the system design process, many alternatives need to be evaluated; fast modeling techniques are vital. In this section we discuss a graphical model description language developed to facilitate model definition and provide the basis for automatic model generation. We also discuss two automated model building packages, one for discrete event simulation models, the other for analytic queueing network models.

3.2.1 GRAPHICAL MODEL REPRESENTATION. One of the most difficult tasks in modeling complex systems is producing a complete model definition. Often the modeler and the design engineer don't speak the same technical language. To facilitate communications between engineer and modeler, and to provide an unambiguous model definition, we have developed a graphical model description language called QNETS.

QNETS diagrams are flowcharts with symbols representing concepts such as queues, delays, or forks. Each symbol belongs to one of five classes. Each class of symbol is identified by a particular shape, as illustrated in Figure 3.2.

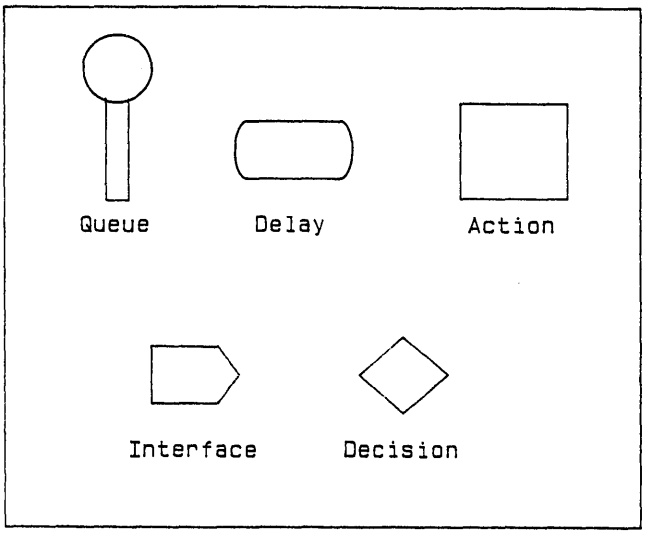


Figure 3.2. QNETS Symbol Classes

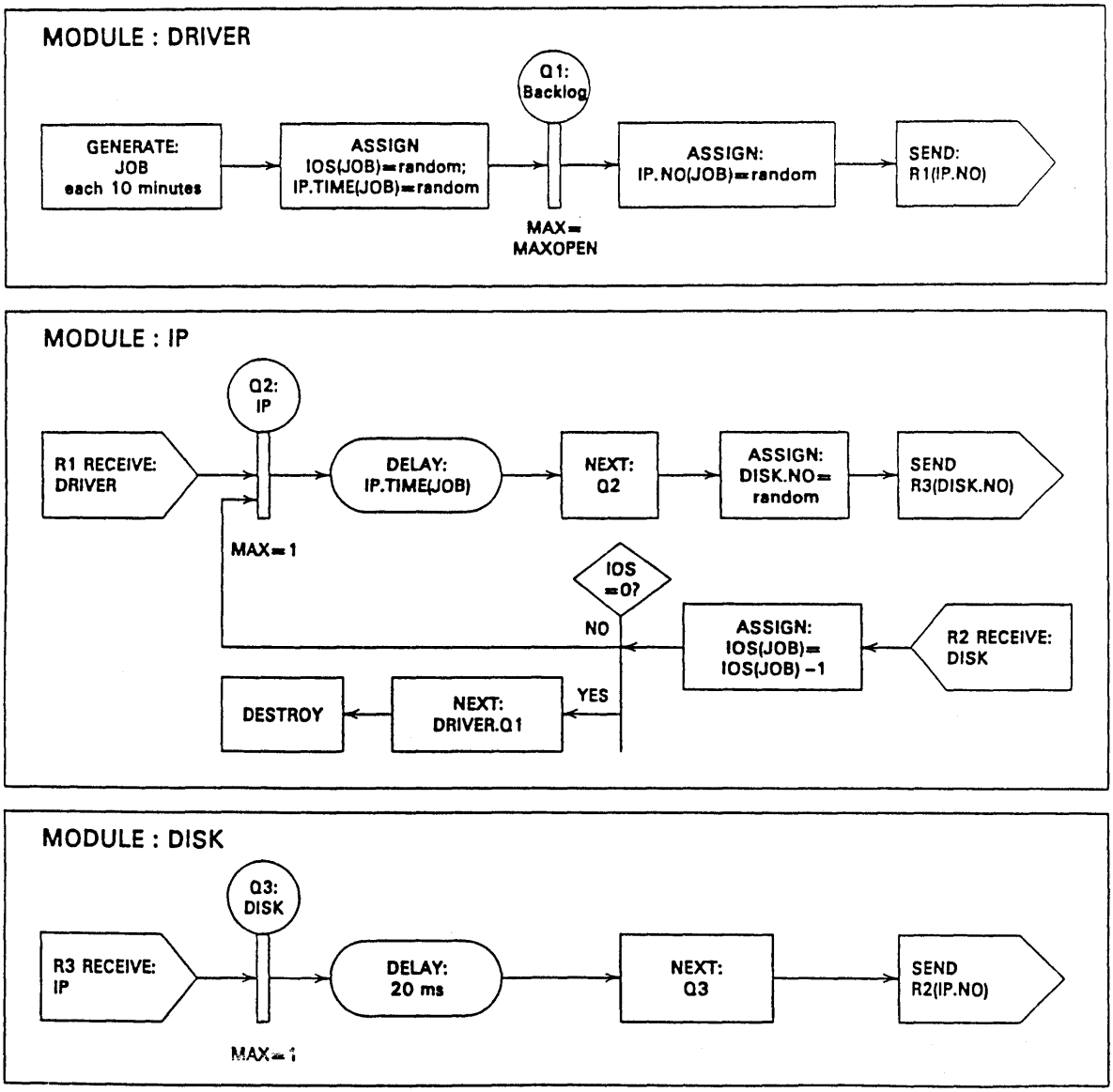


Figure 3.3. QNETS Diagram of a Batch Processing System

Logical divisions of a model, called modules, are described by independent QNETS diagrams. For instance, in a computer system separate modules are constructed for instruction processors, input/output processors, and memory units. Modules may be replicated any number of times and may be nested to any level. This encourages hierarchical, modular model definition and facilitates the use of previously defined modules in different models.

The basic concept of QNETS is the flow of tokens through a network. Tokens carry information from one point in a network to another. While tokens flow along arcs and through action, decision, and interface symbols in zero time, they are delayed at queueing and delay class symbols.

Figure 3.3 shows a simple QNETS diagram of a batch processing system. It contains three module types: DRIVER, IP, and DISK.

This model represents an open system, with batch jobs entering the system at the rate of one every ten minutes. The Generate symbol in the DRIVER module introduces a new token of type JOB every ten minutes. Note that this does not represent the standard open workload. The maximum number of batch jobs open is limited by the maximum concurrency at Q1. Each job, before terminating at the DESTROY symbol, allows a new job to begin (NEXT: DRIVER.Q1). If Q1 is never empty, this represents the classic closed workload.

Each job in the batch system above must first gain access to an IP for a specified time, then frees the IP ("NEXT Q2" symbol) and queues for a disk (selected according to a probability distribution). After releasing the disk, the token returns to the IP module and queues again for the IP. This process repeats until the job has completed its designated number of I/O operations. When all I/O operations are complete, the JOB token releases another JOB token from backlog ("NEXT Q1" symbol) and leaves the system (is DESTROYED).

The number of instances of each type of module is not specified in the QNETS diagram, but is an input parameter to the simulation. Thus the number of IPs and the size of the I/O complex can be varied easily. The SEND and RECEIVE symbols represent the logic for routing the tokens to the correct instance of the module.

3.2.2 **INTERACTIVE MODELING FACILITY.** The construction of a discrete event simulation model can be a long, error-prone task. To overcome this obstacle, we developed the Interactive Modeling Facility (IMF). IMF is a PC-based package which enables the modeler to enter the model definition interactively by constructing QNETS diagrams with a mouse and providing auxiliary information via pop-up menus. IMF ensures the completeness and basic consistency of the model definition, and prompts the user for more information when required. It then generates the complete model source code, including code allowing interactive trace and debug using the

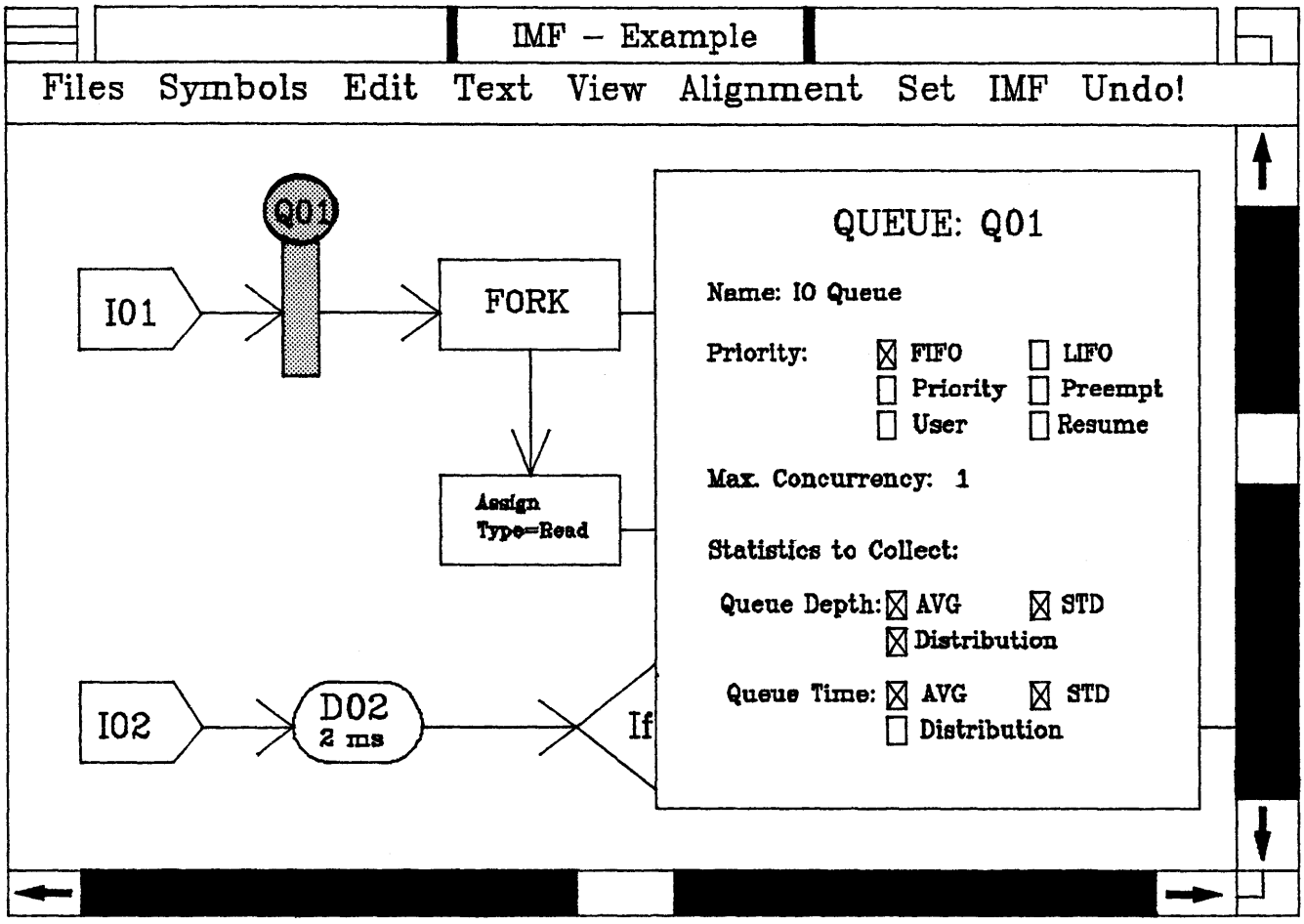


Figure 3.4. IMF Example

names the modeler selected for his modules, tokens, and variables. Models that may previously have taken weeks to develop and debug can, with IMF, be generated in a few hours.

The QNETS editor used to define the model also allows the model to be updated in a matter of minutes. This gives the analyst the ability to quickly study many alternatives for feasibility and design trade-off studies. Figure 3.4 illustrates the interactive model definition provided by IMF.

3.2.3 MVAP. The Mean Value Approximation Package (MVAP) is a general-purpose analytic queueing network package with enhancements for modeling computer systems. It features heuristic one-step mean value approximation solving algorithms, a discrete event simulator, and a simplified exact solver.

The advantages of modeling with MVAP include:

- o Fast and Efficient Solutions
- o Interactive Model Definition
- o Extensive Capabilities
  - Standard Queue Networks
  - Priority Servers
  - Preemptive Priority Servers
  - Priority and Preemptive Priority multi-servers
  - Limited Memory

MVAP is typically used for a .first look at many modeling problems, since we can build an MVAP model and obtain results in a matter of minutes. The results help show whether there is sufficient utilization and complexity to justify the effort of a simulation model, or if an analytical approach is adequate.

As queueing network packages cannot determine the distributions associated with the mean values they estimate, many applications cannot benefit from the use of these fast modeling techniques. In particular, design trade-off studies that require distributions for utilization, throughput, and response time require discrete simulation.

4 HIERARCHICAL APPROACH TO SYSTEM PERFORMANCE ANALYSIS

Our hierarchical approach consists of identifying sub-models by functionality and by the time granularity of related hardware components. At the bottom of the hierarchy are the sub-models having time units in the range from a few nanoseconds to single system clock cycle. At the top of the hierarchy are those with time units in the millisecond to second range. Each model at a given level builds on the results from sub-models at lower levels as the lower-level model results become available.

There are several reasons for using this hierarchical approach. First, it allows each component to be modeled at the appropriate level of detail. Design trade-off studies can be done at the component level and provide the performance data necessary for models further up the granularity dimension. Second, it allows simultaneous model definition and development of the sub-models. Third, high-level models can be constructed and executed using preliminary data even before detailed results from sub-models are available. As the sub-model designs take shape, successively more refined performance estimates become available as input to the higher levels.

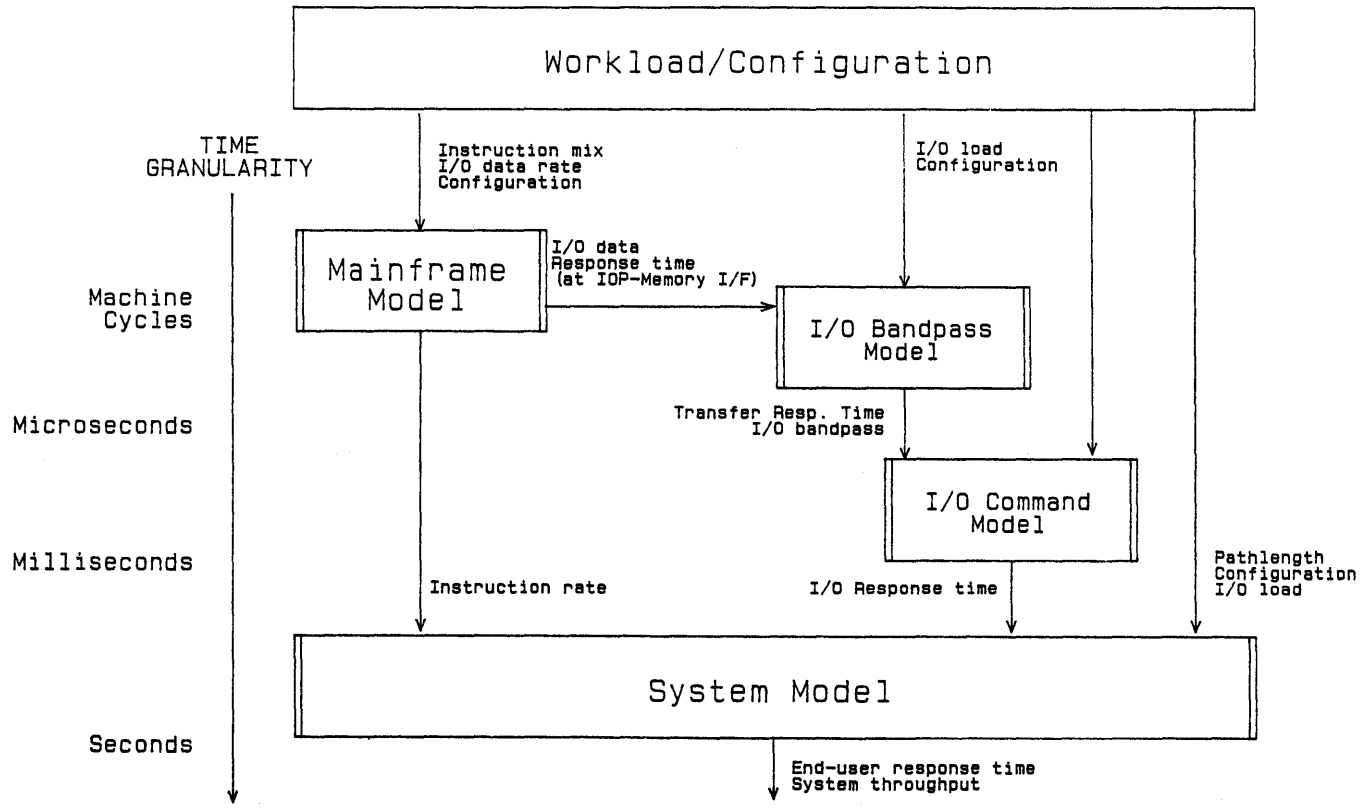


Figure 4.1. Hierarchical Modeling Example

A system-level model at the top of the hierarchy illustrates this well. When a system is at its first stage, design feasibility, we obtain educated guesses and assumptions about the performance of the instruction and I/O processors, storage organization, and mass storage projected to the time of system introduction. With this information, and workload pathlengths and I/O quantity and size, we can obtain preliminary system performance estimates: throughput and response time for given I/O and mainframe configurations. This process helps establish performance requirements at each of the sub-model levels. Conversely, as results are obtained from each of the sub-models, the system-level model uses them to obtain more refined system-level performance predictions.

Figure 4.1 presents a hierarchical modeling matrix based on a time granularity and functional characteristics dimensions. In this section we will first describe the preliminary modeling steps we use. Secondly, we will describe the sub-models in Figure 4.1.

4.1 PRELIMINARY MODELING STEPS

The hierarchy of sub-models is defined by discussing with the designers the questions and concerns they want to see addressed by the models and which areas are likely to change. Once sub-models are identified, three preliminary steps are performed: documenting model descriptions, determining driver requirements, and choosing the appropriate modeling tool.

4.1.1 DEVELOP MODEL QNETS DIAGRAMS. The first step in developing a performance model is for the analyst to work with the designers to develop the QNETS definition of the model. By interacting directly with the designers, design trade-off considerations are incorporated into the model from the start. It is the analyst's responsibility to ensure that enough detail is put into the model so that it can provide the data required by other models higher up the model hierarchy.

4.1.2 DETERMINE DRIVER REQUIREMENTS. As the model description is being developed, one must also determine the workload characterization required as input to the model. Input requirements depend on the type of system or component being modeled, and on the level of detail represented in the model.

For many types of models, the input consists of probability distributions which describe the workload in terms of the type and frequency of events influencing the performance of the system being modeled. When modeling results are highly sensitive to workload probability distributions, the distributions must have a high degree of accuracy to ensure representative results. The distribution of I/O data transfer sizes is an example of an input probability distribution. If the system being modeled is sensitive to sequences of long data transfers, then specifying the average and standard deviation of the transfer size may not be sufficient. A more detailed probability distribution, taking into consideration bursts of activity, may be required. On the other hand, if a model shows little sensitivity to the workload probability distributions, then more tractable distributions and simpler representations can be used in the model.

4.1.3 DISCRETE EVENT SIMULATION OR ANALYTIC IMPLEMENTATION? Several factors must be weighed when choosing between analytic queueing techniques and discrete-event simulation for a particular model. These factors include: the detail required of the input, the complexity of the model, and the types of output statistics required.

Analytic queueing models require the assumption of exponential inter-arrival and service-time distributions. If these assumptions are too restrictive, then discrete-event simulation is necessary.

There are several modeling constructs that do not have exact analytic solutions, and for which analytic approximations may give inaccurate results. These include forking (parallel activities), complex queue disciplines and service-time distributions, and holding multiple resources simultaneously. Models that require detail of such activities must use discrete techniques.

The output of most analytic models is limited to mean values of service times, utilizations, and response times. If designers are concerned about distributions around the mean values, then again, simulation is the choice.

With all that simulation has going for it, it unfortunately tends to be very expensive in terms of personnel and computer resources. For this reason alone, analytic methods are reviewed and used wherever possible.

4.2 HIERARCHY OF MODELS

4.2.1 MAINFRAME MODELS. A mainframe model consists of representations of the instruction processors, portions of the I/O complex, the storage organization, and the interfaces between them (see Fig 4.2). It does not explicitly represent peripherals and communication equipment; rather it represents the load they exert on the main storage complex. The complexity of a module varies according to how much is already known about the component it represents and whether or not design trade-off simulations are foreseen for that component.

Typically, the instruction processor is simulated at the memory reference level. It is often driven with an instruction trace (obtained using a hardware monitor) specifying the types and sequence of memory references. In detailed processor models, these traces can be used directly to model the execution and performance of the processor executing this mix. Alternatively, the instruction sequences can be analyzed to produce shorter, statistically valid instruction streams characterized by instruction class (such as load, store, jump, arithmetic, string), number and type of operands, and instruction execution time. The complexity and run time of the model is reduced by not simulating each individual instruction, yet maintaining important details such as inter-request time and request type distributions.

The processor local cache memory is simulated by a probability distribution resulting in the correct proportion of cache hits and misses for requests generated by the processor module. The distribution is obtained from hardware cache simulations using BIC (see 2.3.3).

The I/O in mainframe models is generally represented by a simple I/O request generator which issues a realistic series of memory references to the main storage unit. However, more detailed representation, including the simulation of buffers, buses, and/or I/O processors may be appropriate in some mainframe simulations. I/O memory requests are generated at rates taken from the I/O profile statistics for each given workload. Sensitivity studies are conducted to determine how significant increases in the I/O memory request rate affects mainframe performance.

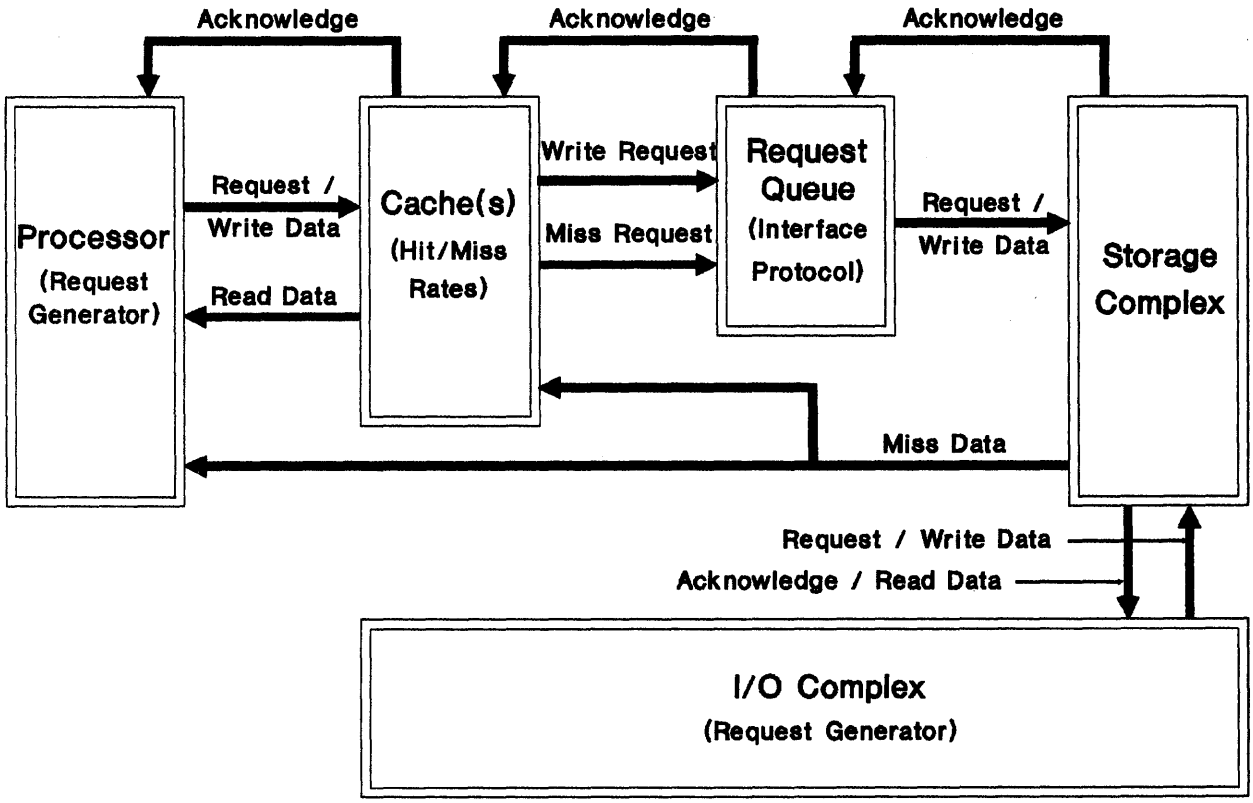


Figure 4.2. Conceptual Mainframe Model

The memory interface is modeled as one or more queues of memory read and write requests generated from the processor/cache modules and the I/O modules. These queues are usually released by the simulated acknowledges or data returned from the memory module. In bus-structured systems, the bus is generally modeled as a separate module.

Complexity of representation of the storage organization varies, depending on whether or not the storage complex contains caches, how sophisticated its queueing mechanisms are, etc. In a simple system, where the main storage unit merely services incoming requests without any of the complexities, the memory may be represented by a queue and a couple of simple delays for access and service time. On the other hand, if the storage complex consists of a hierarchy of real memories, the representation may be extremely complex.

The unit of time for mainframe models ranges from one clock phase to one clock cycle. The primary output of a mainframe model is the instruction execution rate, measured in millions of instructions per second (MIPS). Other statistics include average queue depths, average wait times, and server utilizations. Mainframe models are usually discrete time simulation models; thus, we can obtain standard deviations and distributions of the same parameters.

4.2.2 I/O DATA TRANSFER MODELS. I/O data transfer models are used to study maximum, minimum, and average channel transfer rates. In order to determine aggregate transfer rates, simulations of the data path from subsystem control unit to memory are performed at a level of detail encompassing the transfer of single words along the data path.

The purpose of these simulations is two-fold. The first is to identify which components are bottlenecks of the data transfer path; primary areas of interest are buffer, microprocessor, and bus utilizations. The second purpose is to provide average transfer rates for use as input to I/O request throughput models. Results are given for each combination of channels transferring data concurrently. Delays associated with IOP access to the data path for control purposes are also obtained from these simulations.

I/O data transfer models exhibit a large number of possible states. Along one dimension are differing combinations of active channels. Along another are the combinations of read and write operations. A key modeling consideration is the ease in running the numerous combinations required.

4.2.3 I/O COMMAND THROUGHPUT AND SUBSYSTEM MODELS. I/O processors (IOPs) perform all tasks related to the initiation and termination of I/O requests. They are microprogram-controlled general purpose processors designed to off-load I/O tasks from the instruction processors. IOP models are used to perform design trade-off studies and to determine what types of channels and subsystem configurations these processors can support.

IOP models build on the results of I/O data transfer models by simulating channel program execution around the data transfer operation. Depending on the design, the IOP may or may not combine the control and data transfer functions under a single controlling function. In the latter case, the IOP is not involved directly with data transfer, but it contends with transfer operations for memory and bus accesses. Queueing delays obtained from the data transfer model are

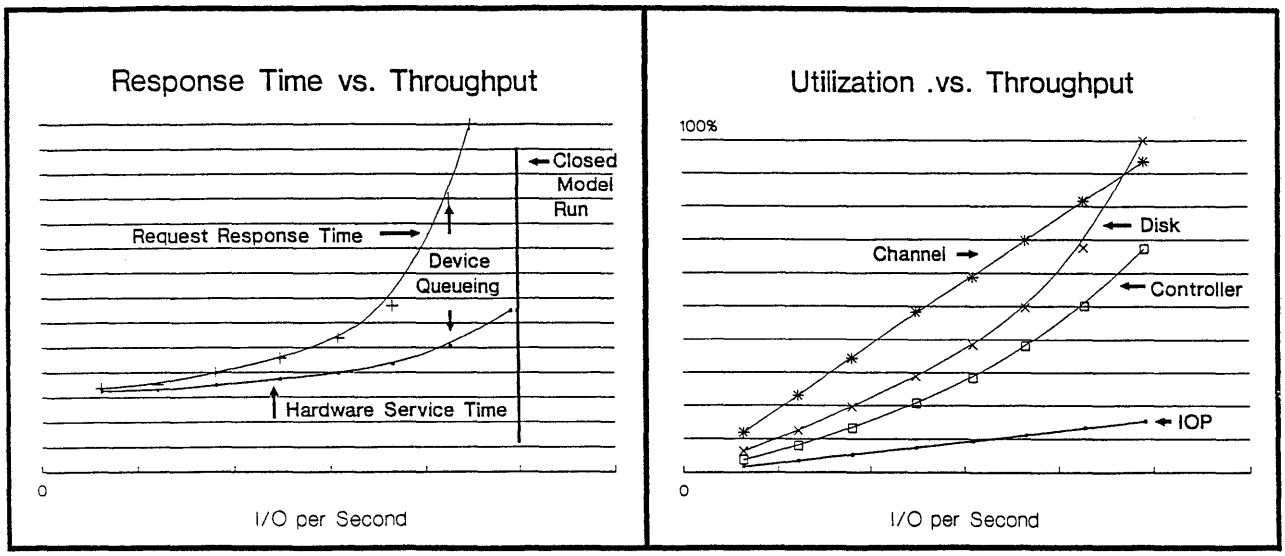


Figure 4.3. Sample I/O Throughput Results

incorporated to represent contention that the IOP experiences when fetching channel control words from memory or transferring control parameters to subsystem controllers.

I/O systems support multiple channel interfaces, each capable of supporting three general types of mass storage subsystems: cache disk, solid-state storage devices (SSD) and disk-only subsystems. By varying timing parameters and cache hit rates (0% for disk-only, 100% for SSD, and somewhere in-between for cache disk), all three subsystems can be represented by a single IMF module.

I/O throughput model results are primarily reported as performance curves in which throughput is plotted against response time (see Figure 4.3). Other results include IOP, channel, control unit, and device utilizations.

4.2.4 MEMORY-MANAGEMENT MODELS. The primary purpose of studying paged systems is to determine what the working set sizes and page fault rates will be for different management schemes. For these studies, a trace of each memory reference could be obtained and used to drive a simulator. As many have found before, that is a very expensive approach. Since we are not interested in individual references, but rather in working sets, we use a somewhat different approach. (We find that this method of tracing is at least 10 times faster than software trace methods and allows us to collect more data on a larger set of programs.) Using BIC in Mode 2 (see 2.3.3), we obtain traces of the activity of a small post-store cache. BIC accumulates the number of references to each block while it is resident in the cache. When flushed from the cache, a record containing the block address and reference count is written to tape. The cache is designed to force frequent block replacements in order to minimize accuracy loss. This is done by having the block size considerably smaller than the page size, specifying a maximum reference count between record flushes, and ensuring that the cache is small enough so that blocks are replaced every 5-10 references on the average. The BIC trace is then used as input to a simulation program designed to study the effects of various memory management algorithms on fault rates and working set sizes of a single program.

We are currently developing a stochastic representation of how well a program runs (number of page faults) for various algorithms, to be used as input to a system-level memory management model. The output of this model will be the system page-fault rate and swap rate, and the associated I/O load. This information on I/O due to memory management can then be input to the system level model.

4.2.5 SYSTEM MODELS. For modeling purposes, system-level workloads are characterized by probability distributions of the following:

Transaction arrival rate (transaction environment)
Number of active terminals and think time (interactive environment)
Number of jobs active (batch environment)
Pathlength per system interaction
Memory size requirements per system interaction
I/O profile (as described in Section 2)

System modeling allows us to build on the lower-level modeling previously described. Given the appropriate assumptions about the probability distributions, system modeling also lends itself well to the standard analytic queueing network modeling techniques. The components represented in the system model are instruction processing, the environment-generated I/O processing, the overhead associated with program loading and task switching, and communications overhead.

The instruction processor is treated as a simple queue which has a total service time equal to the product of the average instruction time and the pathlength; the service time per visit is a function of the total service time, the pathlength between I/O requests, and time slice allotment. The average instruction execution time is that found by the mainframe model; this time is dependent on the benchmark and the configuration. The pathlength is obtained from hardware monitor analysis of the workload during benchmark execution.

In our performance analysis of banked Unisys 1100 Series systems, we have assumed there is sufficient memory to load the programs and to prevent excessive swapping. Thus, the overhead of program loading and

task switching has been lumped into the environment-specific processor and I/O load. In a paged environment, the overhead of page faults is added as well.

There are several methods of modeling I/O processing at the system level. The method selected is based on the information available about the system's interactions with the subsystems. The simplest method basically treats the IOP, channels, control units, and disks as a single server per disk. This allows us to define some variation of the standard central server model and appropriately model the system via analytic queueing network techniques.

Another method treats the IOP as a simple queue with a service time representing the time to perform the command related functions. The I/O subsystems are represented in a somewhat simplified manner, but do take into consideration the overlap in time of device, control unit, and channel usage. The results from these simplified versions allow us to determine optimum system configurations. Subchannel statistics from these models can be compared for accuracy with simulated representations of subsystems.

The results from the IOP command throughput models are used in system modeling studies as results become available. The performance curves for the IOP replace the representation of the I/O complex at the system level.

With communications processors, we also use the sub-model concept to simplify our system modeling. Communication processors, systems in their own right, are modeled by the communications design group to determine response time and throughput. We use these parameters in the system model.

The outputs of the system model are: response time, throughput, utilization, and queue depth. Their values help determine the number of IOPs and channels that should be designed into the system. The ramifications of this extend into physical layout, interfaces, etc.

As the product approaches delivery, the system model is used as an aid to determine whether the product will deliver expected performance. During the product performance validation process, the models are used in conjunction with actual product instrumentation to identify potential problem areas. As part of the validation effort, the models are calibrated to the actual product.

Frequently, the system model does not perish with the release of the product. Instead, the model or its results are turned over to marketing for use as a sizing tool for prospective customers. System models are also used by the designers to evaluate potential performance improvements, future enhancements, and in some cases, next generation products.

5 CONCLUSION

Getting technology and design to coincide with a market window is difficult, and every delay has a potential impact on the product's success. Thus, the ability to provide prompt and accurate evaluation of proposed designs is a paramount ingredient to the product's success. This is the basis for the evolution and development of the performance analysis tools and hierarchical modeling methodology described above.

The hierarchical methodology allows us to break the system into smaller parts; this not only reduces the complexity of the modeling, but also facilitates faster response to performance related design questions by distributing tasks by sub-model. When the inevitable design changes occur, it is also easier and less error-prone to modify a smaller model. As a result we can reduce the time required to respond to changes.

The automated modeling means described above simplifies and expedites the modeling process. By use of the hierarchical methodology and the automated modeling tools, we have reduced the time to respond to design questions from months to weeks and sometimes hours.

Of the above described means of collecting data that characterizes a workload, the hardware simulator (BIC) has proved invaluable in obtaining accurate, detailed workload characteristics. It has been especially useful in obtaining memory referencing patterns used in the study of storage organizations. The data obtained via this technique has been used directly by the designers themselves to answer their design questions; this further expedites the design process.

In order to make sure our methodology is working, and to add to the credibility of modeling, verification is a vital part of our methodology. Not only does verification provide feedback which is vital to the modeler, but also frequently helps the project and the customer by finding performance quirks before introduction. In the case of one of our recently introduced products, the system performance estimates made by these methods were found to be within 2% of the actual measurements, illustrating the success of our methodology.

3090-CLASS PROCESSOR ARCHITECTURES AND WORKLOAD PERFORMANCE LEVELS

Gary Keeney and Greg Simmons
National Advanced Systems

ABSTRACT

This tutorial presents the specifications and fundamental architectures of the IBM 3090 and 308X, Amdahl 5890 and NAS XL series of processors. The presentation includes a discussion of the primary characteristics of several workload types and how they would take advantage of the features of the different processor architectures.

1. INTRODUCTION

IBM announced the 3090 Processor with a range of performance improvements over the 308X series depending on the workload type and the operating system environment. In similiar fashion, Amdahl's 5890 and National Advanced Systems' (NAS) XL series of processors were introduced with workload dependent performance ranges. Although, these three processors provide IBM program and peripheral compatibility, each has its own unique design and internal architecture which offer performance advantages for various software characteristics of the evaluation workload.

Computer architecture concepts and particuliar specifications of these 3090-class processors and their impact on performance can be demonstrated in the framework of measurement methodologies. Performance achievement is dependent on several quantifiable essential workload characteristics. These characteristics when matched with a particuliar processor design can be used to predict the relative performance of each workload type.

2. PROCESSOR CYCLES

An instruction is processed in a number of steps with each step basically requiring one machine cycle: therefore, the speed of any processor is a function of both the machine cycle time and the number of cycles per instruction:

Rate = (1/cycle time) x (1/(cycles/instruction))

MVS Extended Architecture (XA) instructions can be classified as one of three types depending on their memory reference characteristics. Register to register (RR) instructions have no memory reference, storage to register (RX and RS) have one memory access, and storage to storage (SS) instructions have a mininum of two operand fetches and one reference for operand store of the result. The number of memory references is a major determining factor in the number of cycles required to complete instruction execution.

The Arithmetic and Logic unit (ALU) of a processor consists of an Instruction Unit (I) and an Execution Unit (E). The I unit fetches instructions into buffers where the E unit then performs the operation.

The sequence of the E unit is accomplished in stages:

* instruction decode (D)
* operand address generation (A)
* operand fetch (F)
* instruction execution (X)
* operand store (S)

Since the operation of the I unit is overlapped with E unit processing, E unit cycles are the primary factor in processor performance. Generally, I unit instruction fetch can occur during idle periods of E unit operand fetch so that cache contention is minimized.

The relationship between the instruction stages and processor cycles varies by processor design. Each processor offers some unique architectural enhancements in order to decrease the cycles required per instruction.

3. INFINITE CACHE RATE

An elementary method of processor performance rating is the Infinite Cache Rate (ICR). Since this method assumes an infinite cache size with no cache misses, the effect of main memory overhead is ignored.

ICR methodology is basically a function of three primary factors:

* cycle time
* cache bandwidth
* pipeline efficiency

with the latter two influencing the cycles per instruction ratio.

The cycle time is constant for each processor series with the exception of the 3090 E enhancement.

Cache bandwidth is the rate at which data can be read from or written to the cache buffer. The bandwidth depends upon the processor cycles required to fetch a standard operand data quantum of sixteen bytes. The cache bandwidth measurement is a product of:

* data path width
* bank interleaving
* access time

Efficiency of the pipeline depends on many design factors including:

* STO/TLB hit rate
* parallel operand fetch
* multiplicity of execution units
* operand conflict
* instruction modification
* branching

A large STO/TLB hit rate provides less Dynamic Address Translation (DAT) and decreases the fetch requirements for segment and page table entries. Parallel operand fetch decreases the cycles required for two operand instructions by providing the concurrent fetch of both operands. Multiple execution units allow the E stage overlap of two instructions. Operand conflict occurs when an instruction requires the results of the preceeding instruction as one of its arguements. This conflict along with instruction modification and branching all cause pipeline turbulence with a resulting decrease in the execution rate.

4. INTERNAL EXECUTION RATE

A more comprehensive method of measurement is Internal Execution Rate (IER) where given a realistic limited cache size the effects of memory overhead are considered. IER methodology is defined with an infinite amount of main memory so that the effects of paging and data I/O are ignored. Three additional performance factors are introduced with IER:

* cache hit rate
* cache architecture
* memory bandwidth

Cache hit rate depends on both the size of the cache and the aggregate working set size of the evaluation workload.

These processors have different cache architectures. Store-in design provides for the memory store of modified cache data upon cache line reuse. Store-through design does memory store immediately with cache modification so that corresponding memory and cache data always match: however, store-in architecture provides less overhead due to possible multiple data modifications prior to cache line reuse.

As with cache, memory bandwidth is a function of the data path width, bank interleaving and access time. One unique design introduces a memory pipelining technique and a two-level memory hierarchy. Its effect is to increase the effective memory bandwidth and reduce the cycles per instruction ratio particularly for memory intensive workloads.

5. INTERNAL THROUGHPUT RATE

The most inclusive measurement of the performance of a workload is the Internal Throughput Rate (ITR) where the real world effects of a limited main memory with paging and data I/O are included.

ITR requires consideration of two additional design criteria:

* Expanded Storage (ES)
* external I/O processor

ES impact is a function of the memory hit rate (i.e. demand paging) and the ES bandwidth. The influence of external I/O processor design depends on the extent to which I/O path selection and queueing are offloaded from the ALU.

6. WORKLOAD CHARACTERISTICS

The task in matching workload characteristics against processor specifications is to eliminate those elements that have a marginal effect on performance. There are two areas of processor design where performance is greatly influenced by workload criteria:

* pipeline efficiency
* memory overhead

The efficiency of pipeline execution is chiefly impacted by operand conflict and the locality of operand references. Data found in the same cache line as previous references reduces the probability of pipeline wait on a cache miss. The ordering of program data and the sequence of operations on that data have a significant impact on pipeline throughput.

Memory overhead may be reduced with increased cache and memory hit rates. Workload variables determine the number and distribution of memory accesses. These rates are mostly dependent on the instruction type mixture and the aggregate working set size of the workload.

Given values for these essential parameters for an evaluation workload, the relative processor performance may be predicted. Examples will be presented to illustrate the fit of workload characteristics to processor architecture.

REFERENCES:

(1) Gorsline, G.W., Computer Organization: Hardware/Software. Englewood Cliffs, NJ: Prentice-Hall, Inc. 1986.

(2) Smith, Alan Jay, "Cache Memories," ACM Computing Surveys (September 1982).

(3) Tucker, S. G., "The IBM 3090 system: An overview," IBM Systems Journal (Vol. 25, No. 1, 1986).

(4) IBM System/370 Extended Architecture Principles of Operation. IBM Corporation, 1983 (Publication Number SA22-7085).

COUPLED SYSTEMS FOR PERFORMANCE

Dr. Edward I. Cohen•
International Business Machines Corporation (*)
P.O. Box 390, Poughkeepsie, NY 12602
(914)-435-7898

ABSTRACT

A key approach to meeting the requirement for ever increasing performance capacity is the coupling of processors. Tightly coupled multiprocessors provide a wide range of performance levels within a single system image. Many design considerations are important factors in obtaining efficient execution in a large capacity multiprocessing system. These design factors include both hardware and software and in many cases are sensitive to total system capacity as well as the degree of multiprocessing. Providing a balanced system configuration is also key in obtaining the full performance potential of a large processing complex. Many multiprocessor performance characteristics have close analogies in the area of looslely coupled system complexes, and many of the same questions arise in both contexts. Current system structures and requirements lead to a view of future trends and directions in the area of coupled systems.

OUTLINE

PERFORMANCE DEFINITIONS

BALANCED SYSTEMS AND THE 3090E

MP PERFORMANCE ANALYSIS

HARDWARE/SOFTWARE MP INTERACTIONS

LARGE SYSTEMS EFFECTS

LOOSELY COUPLED SYSTEMS

FUTURE TRENDS

<1> CMG 87 EIC 12/87

This presentation begins with some basic standard definitions of system performance, including both responsiveness and throughput. A key concept in obtaining the maximum capacity of a system is maintaining a balanced system configuration. The major components of a balanced system are highlighted with special emphasis on storage and I/O. An analytical approach to MP performance is described, and aspects of both hardware and software MP effects are considered. The most effective way to analyze the dynamics of a large MP system is to view it as a "system" and to understand the interactions of all hardware and software components. Actual examples of this kind of system analysis and design will be presented. Loosely coupled system complexes have their own unique multisystem characteristics and problems. However, in many cases these are very similar to problems encountered within a single MP system. The discussion of these items leads naturally to a comparison of tightly coupled MP and loosely coupled MP from a performance perspective. In conclusion significant trends and directions in the area of large systems are identified and discussed.

(*) Permission is granted to CMG to publish this paper. IBM retains its right to publish this paper elsewhere and distribute copies of it to whomever it chooses.

HOW TO DEFINE PERFORMANCE

RESPONSE TIME FOR ON LINE TRANSACTIONS

ELAPSED TIME FOR BATCH JOBS

EXTERNAL THROUGHPUT RATE (ETR)

- TRANSACTIONS PER ELAPSED SECOND
- SYSTEM CAPACITY IS MAXIMUM ETR

INTERNAL THROUGHPUT RATE (ITR)

- TRANSACTIONS PER CPU BUSY SECOND (AVG.)
- MEASURE EFFICIENCY OF HARDWARE/SOFTWARE COMBINATION
- COMMON MEASURE OF UNCONSTRAINED SYSTEM THROUGHPUT

INSTRUCTION EXECUTION RATE (IER OR MIPS)

- MEASURE OF CPU "HORSEPOWER"
- INSTRUCTIONS EXECUTED PER CPU BUSY SECOND (AVG.)
- SUM INSTRUCTIONS OVER ALL CPs

<2> CMG 87 EIC 12/87

The most critical aspect of system performance for the user and therefore for the system as a whole is responsiveness. This is measured as transaction response time for on-line applications and as elapsed time for batch applications and programs. Fast response time is so critical in the on-line environment that without it other measures of system performance are not very meaningful.

A basic measure of system processing capacity is External Throughput Rate (ETR). It measures the throughput of the system without regard to CPU utilization and when at maximum level is a good indicator of total system capacity. ETR can be constrained by a bottleneck anywhere in the system complex, and therefore it is not a good measure of the processing power of the central processing complex.

Internal Throughput Rate (ITR) measures throughput of the system but considers only the time during which the CPU is actually busy. Therefore, ITR is a standard measure of the efficiency of the central processing complex and its associated software. Accurate ITR values are best obtained in an environment which is free of constraints (storage, channels, control units, DASD arms, etc.). In this way it can provide as clear a picture of processor potential as is possible. Of course the degree of resource constraint existing in a system will affect the actual throughput which is achievable.

Instruction Execution Rate (IER or MIPS) of a processor is simply the rate at which it executes instructions during those periods when it is busy. IER is a basic measure of CPU horsepower which is available to the software. As such, it is an accurate measure of the central processor itself. All other things being equal, the IER ratio of two processors is a good estimate of their relative capacity. However, differences in the efficiency of software and differences in the level of resource constraint can frequently impact the level of performance obtainable. For both ITR and IER, transactions and instructions are summed over all CPs, while busy time is calculated as the average over all CPs.

PROCESSOR IER

IER DETERMINED BY CYCLE TIME & CYCLES PER INSTRUCTION

- CYCLE TIME IS CONSTANT
- CYCLES PER INSTRUCTION (CPI) VARIES GREATLY
  - INSTRUCTION SET
  - INSTRUCTION IMPLEMENTATION
  - INSTRUCTION MIX GENERATED BY "WORKLOAD"
  - NUMBER OF CPs IN TIGHTLY COUPLED MP

EXAMPLE PROCESSOR USES 25 NANOSECOND CYCLE TIME

- ENG/SCI WORKLOAD WITH 2 CYCLES/INSTR ==> 20 MIPS
- MIXED WORKLOAD WITH 3 CYCLES/INSTR ==> 13 MIPS
- DATABASE WORKLOAD WITH 4 CYCLES/INSTR ==> 10 MIPS

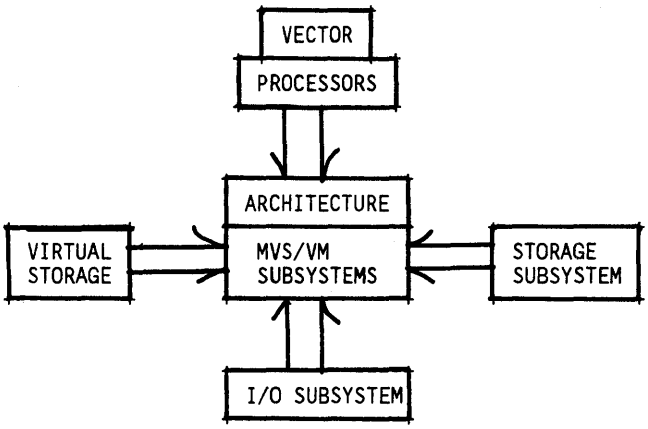
<3> CMG 87 EIC 12/87

The IER of a processor is determined by two factors, the cycle time of the machine and the average number of cycles it takes to execute an instruction. The cycle time is constant for each processor model and is driven by a highly accurate central oscillator. The cycle time achievable in a processor depends on the speed of the technology used, the machine organization, and the distances which internal signals must travel.

The average cycles needed to execute an instruction is highly variable and depends on a great many things, including the specific instruction set architecture used, the method of instruction implementation (degree of microcoded operations), and the number of CPs in a tightly coupled MP. Probably the key factor in determining cycles per instruction (CPI) is the specific instruction mix generated by the workload. In this case workload should be interpreted in the broadest sense to include operating system, subsystems, program products, and application programs. In addition many seemingly unrelated parameters can also impact IER, such as I/O rate, amount of data transferred, and storage usage.

A clear example of this sensitivity is seen when comparing workloads such as data base and engineering/scientific. The CPI in the former case may be about double the latter and as a result the IER in this example would also vary by a factor of two. The key point is that CPI and IER are not fixed numbers uniquely describing a processor. They are really dependent variables which can be affected by a wide range of factors. Because of this, comparative processor measurements must be done in the most controlled and repeatable environment possible to obtain meaningful and accurate results.

BALANCED SYSTEMS CONCEPT



PROCESSORS NEED BALANCED SYSTEM CONFIGURATIONS

3090E FAMILY PROVIDES RICH SET OF CONFIGURATION OPTIONS TO MEET WIDE RANGE OF WORKLOAD REQUIREMENTS

<4> CMG 87 EIC 12/87

In order to be able to utilize the potential power of a large processor a "balanced system configuration" is required. This will provide sufficient resources of each type to meet the needs of the total set of applications which will run on the processor. The key components of a balanced system configuration are the resources necessary to process work, including CPU horsepower (provided by CPs and vector processors), storage for data and programs, I/O capacity including channels, control units, and devices, and also virtual storage. A shortage of any of these resources in relation to the amount needed by the workload will constrain the entire system causing it to operate at less than full capacity, to elongate response times, or both. Each unique application workload requires a specific mix of resources. For example, one workload may require a lot of processor storage and little I/O, while another is I/O bound. A family of processors such as the 3090E must provide for a wide range of requirements by offering a rich set of possible configurations.

3090E CONFIGURATION OPTIONS

6 PROCESSOR CONFIGURATIONS (150E --> 600E)

0 TO 6 VECTOR FACILITIES

32 MB TO 256 MB CENTRAL STORAGE

0 MB TO 1024 MB EXPANDED STORAGE

16 TO 128 CHANNELS

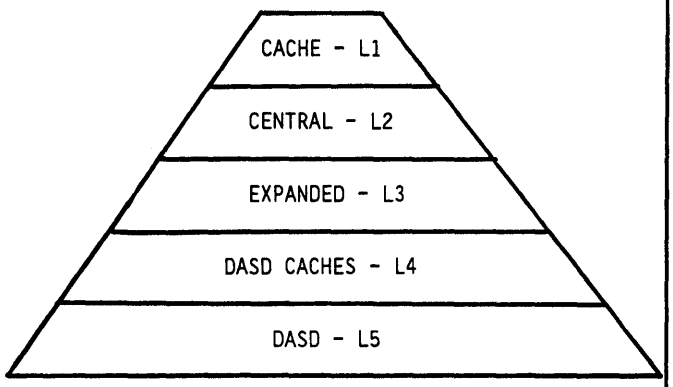
MODEL	VECTOR		CENTRAL		EXPAND		CHANNEL		TOTAL
150E	2	*	2	*	3	*	2	=	24
180E	2	*	2	*	5	*	3	=	60
200E	3	*	2	*	6	*	4	=	144
300E	4	*	2	*	6	*	4	=	192
400E	5	*	2	*	6	*	4	=	240
600E	7	*	2	*	6	*	4	=	336
TOTAL									996

<5> CMG 87 EIC 12/87

The 3090E family of processors offers configuration options covering a wide range of capacity levels and providing multiple options for each level of processing power. There are six models, starting with two uniprocessors up to the largest single image processor available, the 3090-600E. Each model can be configured with from zero to N vector processors where N is the number of CPs. Multiple options and combinations of central storage, expanded storage, and channels are available with each model. The total number of combinations available across the model line is 996 configurations. For each model a maximum configuration of each resource is available which can handle the most stressful load attainable in any production environment. For example, the maximum number of channels offered on a particular model has been determined by studying production workloads and identifying the most stressful workload environment. In this way a 3090E configuration is available which can satisfy almost any conceivable workload requirement.

The strategy behind the design of the storage subsystem of a large processor is called a "Storage Hierarchy." Its goal is to provide multiple storage levels so that the average cost per bit across all storage is very close to the cost of the least expensive level and the average access time to data, as seen by the processor, is close to the fastest level. The hierarchy has many levels, for example L1 (CP cache) to L5 (DASD). In this example both access speed and cost decrease as one moves down the hierarchy. Size increases dramatically from top to bottom along the hierarchy, and this leads to the desired average price per bit.

STORAGE HIERARCHY



SPEED DECREASES GOING DOWN HIERARCHY (L1 -> L5)

COST DECREASES GOING DOWN HIERARCHY

SIZE INCREASES GOING DOWN HIERARCHY

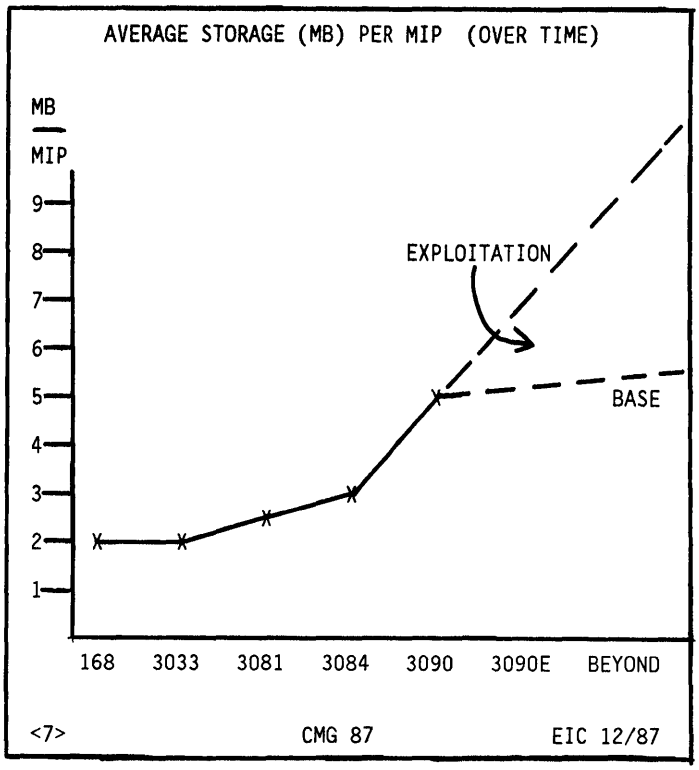
INTERLEVEL MOVES MANAGED BY HARDWARE OR SOFTWARE

GOAL IS "AVERAGE SPEED OF L1" & "AVERAGE COST OF L5"

<6> CMG 87 EIC 12/87

Data movement between levels is normally managed by hardware, microcode, or software. For example, movement between processor cache and central storage is managed by CP hardware using an LRU philosophy. Movement between central and expanded storage is managed transparently by the operating system. Finally, movement between DASD and DASD caches is managed by microcode in the cached DASD controller. While each of these storage managers may be implemented somewhat differently the basic goals and algorithms (LRU) are the same. The concept of a storage hierarchy is key to providing a fast storage subsystem at a reasonable cost and will continue to be the guiding strategy in large system designs.

Historical trends in the usage of storage can be studied by graphing average storage (MB) per MIP over time, where time can be represented by the chronological sequence of IBM high end processors (168, 3033, etc.). Using typical storage configurations provided with those processors we see that during the 168 and 3033 time frame configurations were about 2 MB per MIP. That rose slightly with the 3081 and 3084 processor lines to about 3 MB per MIP and then jumped to about 5 MB per MIP for 3090s. This trend shows that storage configurations were constrained by technology limits and costs until the 3090 provided large storage configurations of central + expanded storage.



The storage available with 3090 and 3090E processors allows one to achieve full system capacity and can relieve the problems of paging/swapping performance and tuning. The interesting question is what will happen in the future? If future use of storage is similar to today's usage one would expect to see a fairly flat curve which would lead to configurations growing only at about the rate of MIPS growth. However, with the attractive and impressive gains being made in storage technology and price, the direction is to exploit larger storage configurations for new functions. A key area of storage usage will be in large database buffer pools. This can reduce database I/Os dramatically, provide improved response times and system ITRs, and allow for somewhat smaller database I/O configurations. Functions such as this which can successfully exploit large storage will move the MB per MIP trend higher. The 3090E systems can provide over one gigabyte of processor storage and are positioned to take advantage of new uses of storage such as large buffers.

The MP/UP throughput ratio depends on the software and hardware factors determining the MP/UP CP MIPS ratio and the change in path lengths between the MP and UP case. In general one expects to get less than N times a UP from an N-way MP system. The same formula can be used to calculate the ratio of any two systems (for example quadratic to dyadic) by using the ratio of 4-way to 2-way in each factor in the equation. By studying the formula one can see that both hardware and software are potential sources of extra processing time on highly parallel and dynamic MP systems.

TIGHTLY COUPLED MP PERFORMANCE

$$Pn = N * (1 - Hn) * (1 - Sn)$$

WHERE

Pn = N-WAY MP/UP ITR RATIO

N = NUMBER OF CPs

Hn = N-WAY MP HARDWARE FACTOR

Sn = N-WAY MP SOFTWARE FACTOR

EXAMPLE: N=2, Hn=5%, Sn=5%

N=4, Hn=10%, Sn=10%

P(DYADIC/UP) = 2 * (.95) * (.95) = 1.805

P(QUAD/DYADIC) = 4/2 * (.90/.95) * (.90/.95) = 1.795

<8> CMG 87 EIC 12/87

While the cycle time remains constant the cycles per instruction (CPI) tends to increase with the number of CPs in the MP system. This is the result of many functions which must be executed differently on an MP, both in hardware and software. A shift in instruction mix is generally observed when comparing an MP to a UP, and this is the result of MP and large system management algorithms executed by software. For example, SIGP instructions are only executed on MPs, and they take a long time to execute compared to more typical average instructions. In addition other instructions and functions may be executed on both UP and MP systems, but their frequency may be higher on MP and the time it takes to execute them may be higher. For example, since supervisory functions in the software are executed more frequently on an MP, privileged ops such as LPSW and LCTL, which are relatively slow, may be executed more frequently per MIP on an MP. Also, an instruction such as Compare & Swap (CS) requires more cycles as CPs are added because N-1 of them have to be synchronized with the issuer of the instruction.

Another area in which MP is affected is in cache performance. Each CP needs to coordinate with all other CPs when locating a valid copy of a particular cache line, since another CP may have a changed copy in its cache which has not been reflected in storage. In this case the CP needs to crosscheck with the other caches. Also, from time to time an active piece of data may be shifted from one CP cache to another before the first has finished using it and this can create data contention.

A final area of focus is access to storage and contention for the access paths and staging facilities. Since an MP generates a higher requirement on storage, the level of contention for common, shared facilities is necessarily higher with the higher levels of MIPS available in the MP. These characteristics have been recognized and studied since the early years of MP design, and sufficient capacity and redundancy is now designed into MP systems to handle peak loading requirements.

The path lengths executed by software increase both with the increase in CPs and as a result of higher levels of

total system MIPS. Certain software functions are used more heavily on a large MP system than on a smaller UP such as dispatching, wait state processing, and locking. In addition more instructions are needed to manage the more dynamic environment found in an MP system. It has been recognized that many functions which at first seem to exhibit MP growth characteristics really are being driven higher by the need to manage a larger system image and its associated larger sets of resources and work units. These "Large System Effects" have been identified and removed on a regular basis in new software releases and will continue to be studied. These effects react solely to the level of capacity independent of the number of CPs configured. Since MPs have always been the largest image available these effects are always observed first on the MP. Large system effects impact both IER and path lengths since they typically do not execute as efficiently as average instructions. Overall, these MP and large system characteristics are well understood, and each new release of software and each new processor model incorporates the latest improvements in these areas.

MP HARDWARE, SOFTWARE, AND SYSTEM FACTORS

CYCLES PER INSTRUCTION INCREASES WITH NUMBER OF CPs

- INSTRUCTION MIX CHANGES
- CACHE INTERACTIONS
- STORAGE CONTENTION

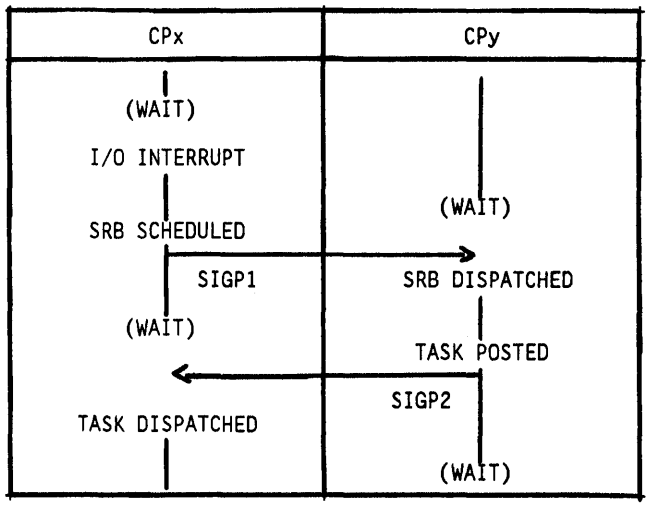
INSTRUCTIONS PER TRAN. INCREASE WITH NUMBER OF CPs

- WAIT STATE PROCESSING
- LOCKING AND RESOURCE UTILIZATION
- MANAGING MULTIPLE PROCESSORS

INSTRUCTIONS PER TRAN. INCREASE WITH TOTAL SYSTEM MIPS

- "LARGE SYSTEM EFFECTS" IN SOFTWARE
- DEPENDS ON WORKLOAD AND CONFIGURATION SIZE
- MOST NOTICEABLE ON LARGEST SUNGLE IMAGE MPs
- IMPACTS BOTH IER AND INSTRUCTION PATH LENGTHS

WAIT STATE PROCESSING



SIGP IMPACTS PATH LENGTH AND IER

FREQUENT CONTEXT SWITCHES IMPACT CACHE LOCALITY

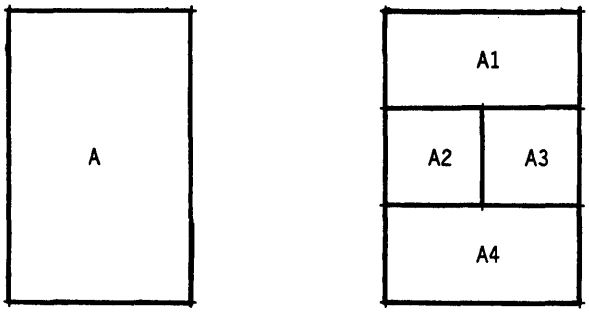
PROBABILITY OF ONE OR MORE WAITING CPs FOR 600E

- 88% AT 70% CPU UTILIZATION
- 74% AT 80% CPU UTILIZATION
- 47% AT 90% CPU UTILIZATION

The management of waiting processors has steadily gained importance as the number of CPs has grown. For example, on a 6-way MP the probability of finding one or more processors in the wait state is 47% even at 90% CPU utilization. The processing of I/O interrupts and the interaction with waiting processors illustrates the increased dynamic nature of N-way MP. After the I/O interrupt is accepted, an SRB is scheduled to run in the address space waiting for the I/O to complete. At this time any waiting processor will be signalled (using a SIGP) to dispatch and execute the SRB while the initial processor goes on with what it had been doing. This uses the philosophy that there is nothing of lower priority than sitting in the wait state. Similarly, when the SRB posts the task as ready to run (since its I/O has completed) the posting function will SIGP any waiting processor to dispatch and execute the task. This process exploits the parallelism available on an MP and optimizes responsiveness. However, it also execute a lot of extra instructions which tend to execute slowly. Wait state processing was thought to be of little importance because interrupting wait state is not like interrupting a busy task. However, its increasing frequency on large MPs indicates that a more efficient mechanism may offer improved performance.

RESOURCE SERIALIZATION AND LOCKING

TIGHTLY COUPLED SYSTEMS REQUIRE MORE PRECISE LOCKING



ATOMIC SERIALIZING INSTRUCTION (CS)

- ALWAYS USED TO OBTAIN LOCKS (MP AND UP)
- REQUIRES MORE CYCLES ON MP TO SYNCHRONIZE N-1 CPs

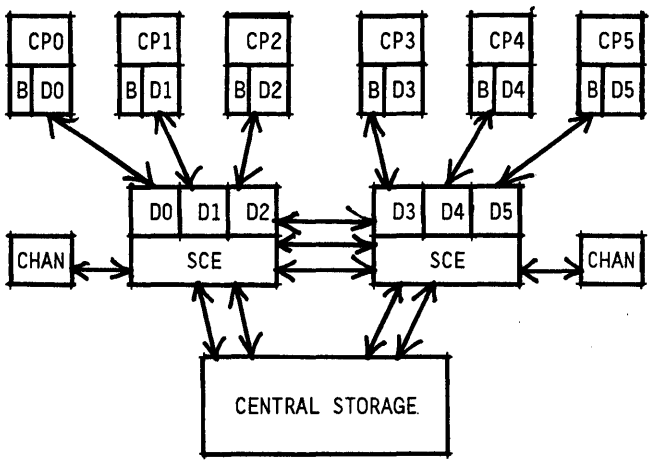
LOCK CONTENTION AFFECTS BOTH IER AND ITR

<11> CMG 87 EIC 12/87

Resource serialization and locking is a basic requirement for any software system which needs to support parallel execution of independent tasks and functions. Serialization using an atomic updating instruction such as Compare & Swap (CS) is used for locking and also to update any shared, common, or global field which may be accessed in parallel. For example, many system level counters maintained within MVS are incremented and decremented using CS. In all of these cases the CS instruction is executed and the lock or counter is serialized even if execution is on a UP. In this case the performance difference in MP is in the timing of the instruction itself, and this is reflective of the extra time it takes to synchronize N-1 other processors.

As software has evolved to be able to support more MIPS and higher levels of MP it has been necessary to enhance the granularity or precision of the locked entities in order to reduce contention. For example, if a large set of data resources (labeled A) are all controlled by a single lock then multiple functions needing access to subsets of A will contend with each other since they each must serialize all of A as a single entity. In this case the enhancement is to subdivide A into parts (A1-A4) which correspond to the subsets frequently accessed together by each of the functions using A. In this way multiple functions (up to four of them) can share the resources which only one could use previously. This approach has been used in many software products through the years to improve MP performance. However, care must be taken in this approach since if A is broken up unwisely, some functions will now need to obtain serialization on multiple resources from the set A1-A4. In this case any reduction in contention may be compromised and multiple lock obtains and releases may be required (using CS). In general, resource and lock contention affects both IER and ITR, and it can also occur on a UP system which uses suspend locks to serialize one task with another in a multiprogramming environment.

STORAGE SUBSYSTEM OVERVIEW



KEY STORAGE COMPONENTS

- SCE AND SCE TO SCE BUSSES
- HIGH SPEED STORE-IN BUFFERS (PROCESSOR CACHES)
- CACHE TO CACHE BUSSES
- CENTRAL STORAGE ACCESS BUSSES AND REGISTERS

<12> CMG 87 EIC 12/87

An overview of the storage subsystem shows that the System Control Element (SCE) is the key component. It controls all accesses to storage and is central in the resolution of all cache faults. For high speed transfers and wide bandwidth there are three busses between SCEs in an MP system. The channel processing engine is also ported to the SCE (as are all the CPs) since it needs to store and fetch data from storage for each I/O. Each SCE is designed to handle the stress load of a fully configured system consisting of requests from three CPs, from the other SCE, and from the channel engine. Each CP has its own high speed cache buffer and associated cache directory. In addition the SCE maintains a duplicate copy of the cache directories in order to resolve crosschecking on cache faults.

The caches in the 3090 and 3090E processors are store-in, which means that a change to a data element in cache does not have to be reflected back to central storage at the time of the update. In this design a changed data element may be valid only in the copy in a cache and the copy in central storage may be obsolete. The SCE manages all of these events so that data integrity is maintained at all times. Although a store-in cache design is more complex than a store-through design it is advantageous since it dramatically lowers the data flow to central storage. When the SCE recognizes the case in which a cache line requested by a processor is in another processor's cache it obtains that line for the requestor via a cache to cache transfer rather than from storage. In the case of a changed line of data this reduces the delay time significantly since a cache to cache transfer is much faster than cache to storage to cache. Even in the case of an unchanged line the cache to cache path is frequently used since it reduces the load on the storage access paths. Overall, this structure is a sophisticated and time proven

design which provides the required function and performance.

STORAGE IMPACT ON PROCESSOR IER

CACHE MISS

- DIRECTORY LOOK UP
- FETCH CACHE LINE FROM CENTRAL STORAGE
- OBTAIN DATA VIA CACHE TO CACHE PATH

CYCLES PER INSTRUCTION (CPI)

- CPI WITH INFINITE CACHE ASSUMPTION
- FINITE CACHE STORAGE ADDS TO CPI
- DEPENDS ON CACHE MISS RATE AND AVERAGE CYCLES NEEDED TO RESOLVE MISS

CYCLES TO RESOLVE MISS

- CACHE TO CACHE TRANSFER
- CONTENTION FOR CENTRAL STORAGE ACCESS PATHS

MP STORAGE DESIGN INCLUDES CONTENTION

- DESIGN POINTS ARE 3-WAY AND 6-WAY MPs
- CONTENTION LESS ON UP, DYADIC, AND QUAD
- CONTENTION ALSO DEPENDS ON CPU UTILIZATION

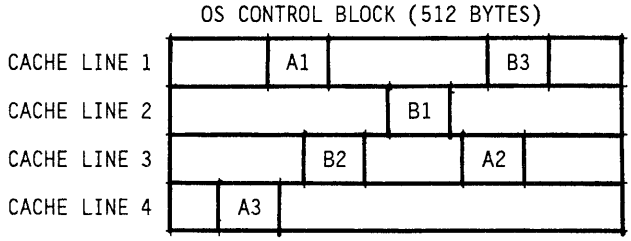
<13> CMG 87 EIC 12/87

The cycles per instruction (CPI) performance of a processor can be studied in two stages. The first stage assumes that all data references are satisfied from cache (100% cache hit is equivalent to having an infinite cache). In this stage the analysis centers around the processor logic normally divided into I-units and E-units. Infinite cache performance is very important since actual cache results typically show that 95+% of all storage references are satisfied from the processor's cache directly. Cache misses are much less frequent and can be analyzed separately. The second stage studies the finite cache impact to CPI which includes the rate of cache faults and the time required for resolution. In this second stage the impact of the storage subsystem is key and both cache and storage delays need to be considered.

When a cache fault is detected the first step is to search for the requested line in other caches. This is done by the SCE using its copies of the cache directories. The first case is when no other cache contains the data. This is a simple miss, a storage fetch is initiated, and the time to resolve the cache fault depends on the contention encountered on the access path to central storage. In the second case, if another cache contains the data and it is changed in that cache, then that is the only valid copy and a transfer of the changed line over the cache to cache path is invoked. In the third case, if another cache holds the line with no change then a copy of the line can be obtained via cache to cache or from storage depending on the type of request. In this case the holder of the line might also retain a copy.

The current storage design point is the 3-way and 6-way MPs because they produce the most stressful requirements on the storage subsystem. Any practical system must be designed with contention detection and resolution in mind since it is a normal occurrence in these situations. Sufficient capacity is designed into these areas to handle maximum loads with reasonable amounts of contention for resources. Of course the level of contention is less when the CPU utilization is lower or when less than the full complement of CPs is active or installed (UP, dyadic, quad).

ANALYSIS OF CACHE PERFORMANCE



SOFTWARE DRIVES CACHE PERFORMANCE

INCREASED CACHE SIZE CAN INCREASE XIs

CPU UTILIZATION AFFECTS CACHE PERFORMANCE

SOPHISTICATED LABORATORY INSTRUMENTATION REQUIRED

- STUDY CACHE MISS AND XI HOTSPOTS
- IDENTIFY EXACT INSTRUCTION AND STORAGE REFERENCE

SOFTWARE REDESIGN GOALS

- LOCALIZE A1-A2-A3 AND B1-B2-B3
- REDUCE DIRECT CACHE MISSES BY A FACTOR OF THREE
- REDUCE CACHE MISSES CAUSED BY CACHE FLUSHING
- REDUCE XIs BETWEEN FUNCTION A AND FUNCTION B

<14> CMG 87 EIC 12/87

Software design, particularly data formats, is the driving force behind cache performance both for cache miss rates and the level of intercache contention. Studies have shown that operating systems have cache hotspots which are specific instructions in frequent software functions which suffer an unusually high rate of cache faults. This is also true for cache interference (cache cross interrogate), but these two sets of hotspots may be disjoint.

In the example control block, two functions A and B are shown which exhibit hotspots on references to fields A1, A2, A3 and B1, B2, B3 respectively. The first problem is that each function requires three of the four cache lines in this control block and will generate many more cache faults than absolutely necessary. Also, an indirect result is the flushing of the cache. This type of control block frequently exists within a queue of identical

blocks, many or all of which need to be accessed. In this case a large fraction of the cache may be replaced by these same three lines multiplied by the number of control blocks on the queue. These functions also can cause a lot of cache interference since they are trying to share the same cache lines (although they are actually using different data fields).

A common solution is to rearrange the control block in the next available release of the software so that function A's data is in one cache line and function B's data is in a different cache line. In this way the cache faults in each function will be reduced by two-thirds, there will be less chance of cache flushing, and contention will be significantly reduced.

LARGE SYSTEM SOFTWARE EFFECTS

SOFTWARE SYSTEM/SUBSYSTEMS INTERNAL DESIGN

- BASIC STRUCTURE IS LINKED QUEUE OF DATA BLOCKS
- KEY ACTIVITY IS SEARCHING QUEUE
- SEARCH PATHLENGTH INCREASES WITH TOTAL SYSTEM MIPS

PROCESSOR POWER	X	2*X
QUEUE LENGTH	A	2*A
INSTRUCTIONS/ELEMENT	B	B
SEARCHES/SECOND	C	2*C
TOTAL INSTRUCTIONS	A*B*C	4*(A*B*C)

HIGHER MIPS LEVELS EXPOSE NEW LARGE SYSTEM EFFECTS

LARGE SYSTEM AND MP SOFTWARE EFFECTS OCCUR TOGETHER

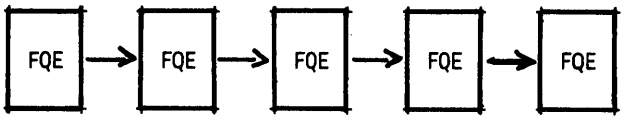
LARGE SYSTEM EFFECTS REMOVED IN NEW SOFTWARE RELEASES

<15> CMG 87 EIC 12/87

In all software systems and subsystems the basic control structure is a linked queue of identical blocks, and the key action against these queues is searching. In almost all cases the search requires more instructions on a more powerful system because the queue tends to be longer in a larger system with more active elements (devices, channels, storage, tasks, etc.). This growth in path length is independent of whether the growth in MIPS is achieved by adding processors or by replacing one processor with a similar but faster model (UP to UP, dyadic to dyadic, etc.).

When the speed doubles, the function is executed twice as often, and each time it must search twice as many elements. This results in a four fold increase in instructions and a two fold increase in time because the instructions will be executed twice as fast. MP effects frequently occur with and because of these large system effects. For example, if the longer search requires serialization, the increased time spent in a serialized state will cause more contention (lock spin). Also, while searching the longer queue more cache flushing can occur. Often the frequency of execution on an MP will accentuate the large system effect. As new levels of MIPS and parallelism are reached, new large system and MP effects are identified and resolved as quickly as possible.

MVS/370 SQA GETMAIN/FREEMAIN



ORDERED BY ADDRESS

RANDOM ORDERING BY SIZE

GETMAIN

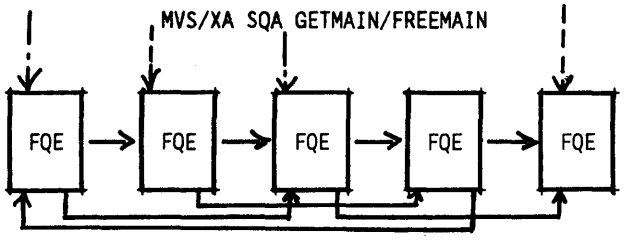
- SEARCH ENDS IF EXACT FIT FOUND
- ELSE ENTIRE QUEUE SEARCHED FOR BEST FIT

FREEMAIN

- QUEUE SEARCHED FOR INSERT POINT (BY ADDRESS)
- RECOMBINE FQEs IF POSSIBLE
- HALF OF QUEUE SEARCHED ON AVERAGE

<16> CMG 87 EIC 12/87

MVS/370 SQA Getmain/Freemain remains the classic example of the intermingling of large system and MP effects in software. In this case the queue of Free Queue Elements (FQEs) represents unassigned areas of SQA storage. The queue is ordered only by ascending address, and therefore the size order of the free elements is totally random. Getmain searches the queue until an exact size match is found; if no match is found it searches until the end of the queue and assigns a piece of the block which is closest in size but larger than the request. Most getmain requests search the entire queue. Freemain searches until the correct insert point is found based on address, and in general somewhat more than half the queue needs to be searched. As MIPS increase, the SQA area increases, the area becomes more fragmented with many small blocks, and the queue grows to enormous length. In addition the search uses a global MVS/370 spin lock to serialize access to the queue. All of MVS/370 storage management must contend with this elongated search algorithm for use of the lock. This case shows a large system effect which is compounded by and feeds a secondary MP effect.



ORDERED BY ADDRESS AND FQE SIZE

INDEX TABLES FOR BOTH ADDRESS AND SIZE

GETMAIN

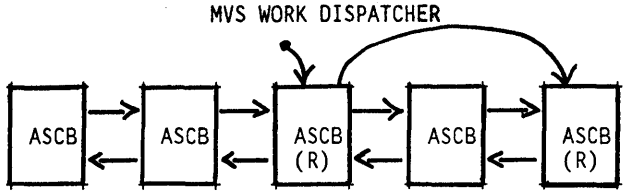
- FIND INDEX FOR SIZE REQUESTED
- SEARCH QUEUE SEGMENT STARTING AT INDEX POINT
- BEST FIT IS FIRST BLOCK LARGE ENOUGH

FREEMAIN

- FIND INDEX FOR ADDRESS RANGE
- SEARCH AND RECOMBINE AS IN MVS/370

<17> CMG 87 EIC 12/87

In MVS/XA the SQA FQE search is reduced by adding additional structure to the queue design. A second set of pointers were added which could be used to search the queue in size order instead of address order. Also, index tables were added for both size and address order which can be used to direct the search to a small subset of the queue. Getmain finds the index for the requested size and searches the queue from that point in ascending size order until the first block large enough is found. Freemain works as before except that the index leads quickly to the correct segment of the queue. These enhancements would be enough to resolve the associated MP lock out, but the locking structure was also enhanced and this function now has its own unique lock. These changes solved both the large system and MP aspects of this problem and led to improved MP/UP IER and ITR ratios even though the hardware involved was not changed at all.



QUEUE OF ALL SWAPPED IN USERS (ASCBs)

QUEUE SEARCHED FOR HIGHEST PRIORITY READY WORK

QUEUE GROWS WITH TOTAL MIPS

ALGORITHM ALSO EXHIBITS MP GROWTH

- HIGHER DISPATCH RATE ON MP
- ALL OTHER QUEUE SEARCHES SERIALIZED BY SPIN LOCK

MAJOR CAUSE OF "LOW UTILIZATION EFFECTS"

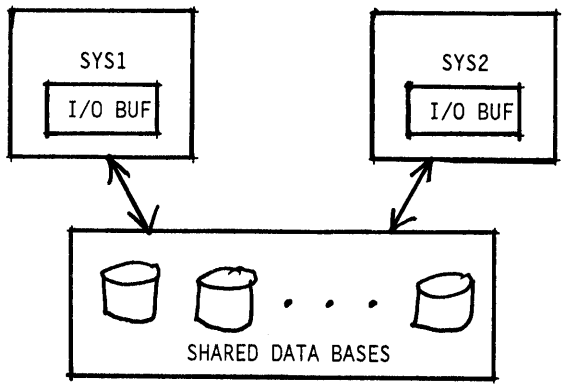
FIXED BY "TRUE READY QUEUE" DISPATCHER IN MVS/SP 2.1.7

<18> CMG 87 EIC 12/87

Another large system and MP problem was recently resolved in release 2.1.7 of MVS. The MVS dispatcher does frequent searches of a queue of all swapped-in address spaces (ASCBs) looking for the highest priority ready work. This queue grows with MIPS and tends to have very few ready ASCBs since most address spaces are in wait state. This algorithm also impacted MP/UP ratios because the dispatch function is invoked more frequently on an MP system. Also, other functions which use the queue needed to serialize it which also was reflected in the MP/UP ratio. This algorithm is the major cause of the "Low Utilization Effect" which uses extra CPU time to manage a system which does not have enough work to keep it very busy (time spent looking for work and processing wait state entries/exits). The "True Ready Queue" dispatcher enhancement fixed this problem by introducing a new set of pointers linking only the ready ASCBs so there should never be a long search for ready work. Again, this affected MPs more since their dispatch rate per transaction is higher and their queues are longer.

These examples show the difficulty in separating MP unique effects from other related effects. In general this requires high precision laboratory instrumentation of a running system to obtain accurate enough detailed information to pinpoint cause and effect relationships correctly.

LOOSELY COUPLED SYSTEMS



BUFFER COHERENCY PROBLEM

- DATA BASES ------> CENTRAL STORAGE
- I/O BUFFERS ------> PROCESSOR CACHES
- BUFFER CASTOUT ------> CACHE TO CACHE XIs

LOOSELY COUPLED PROBLEMS SIMILAR TO MP

<19> CMG 87 EIC 12/87

Frequently, loosely coupled problems are analogous to problems in a tightly coupled MP. For example while MVS has to schedule work between CPs in single image MP system, JES has to do the same thing across multiple systems in a complex. Problems caused by the requirement to share data are also very similar. In single image MP this problem becomes the sharing of data between processor caches with a common central storage. In loosely coupled complexes this problem is reflected as multiple systems trying to share data from a common centralized data base. In this case the general problem of buffer coherency appears as a common problem of any data sharing design.

In trying to compare and contrast the performance of single image and partitioned loosely coupled it is true that single image mode will have lower average CP MIPS and increased path lengths in comparison to a single partition. However, instruction path lengths will also grow in the loosely coupled case due to the overheads of sharing data (using functions such as IRLM and GRS). JES2 and JES3 requires extra instructions when trying to manage and schedule a workload across multiple systems. Also, there may be extra overhead due to increased paging in the partitioned case because of the need to support two copies of the operating system and its subsystems. Finally, a system such as MVS can more efficiently schedule and balance work across tightly coupled CPs than JES can across multiple system images. So, the conclusion is not clearly in favor of one over the other. While the IER and even the ITR may be lower on single image, the processing capacity (max ETR) may be higher.

LOOSELY COUPLED PERFORMANCE

CP MIPS IS LOWER ON SINGLE IMAGE MP

INSTRUCTIONS PER TRAN IS HIGHER ON SINGLE IMAGE MP

INSTRUCTIONS PER TRAN GROWS ON LOOSELY COUPLED

- SHARED DATA FUNCTIONS (IRLM, GRS)
- WORKLOAD BALANCING FUNCTIONS (JES2, JES3)
- INCREASED PAGING FOR STORAGE OF TWO COPIES OF MVS

SINGLE IMAGE MVS CAN MANAGE MP WORKLOADS BETTER THAN MULTISYSTEM JES2/JES3

CONCLUSION ---- NOT CLEARCUT

- IER LOWER ON SINGLE IMAGE
- EVEN ITR MAY BE LOWER ON SINGLE IMAGE

BUT

- PROCESSING CAPACITY (ETR) MAY BE HIGHER

<20> CMG 87 EIC 12/87

In conclusion, future trends have already been established and will continue. Both modes of coupling processors will continue to evolve, and the typical large installation will need to be a loosely coupled complex of large tightly coupled MPs. In the area of performance, single image performance will continue to improve as knowledge gained from working with current 6-way processors is incorporated into new product releases. Loosely coupled function and performance will continue to be enhanced. In the area of balanced systems, multiple configuration options will continue to be needed to meet the diverse needs of a broad spectrum of workloads. I/O capacity will keep pace with the growth in processing capability. Storage will become even more important in providing performance gains as large storage configurations and exploitation become more widespread.

FUTURE TRENDS

COUPLED SYSTEMS

- FUTURE NEEDS TIGHTLY COUPLED MP AND MULTISYSTEM
- COMPLEX OF LOOSELY COUPLED LARGE MPs

PERFORMANCE OF COUPLING

- MP PERFORMANCE STILL KEY FOCUS FOR HIGH END SYSTEMS
- LOOSELY COUPLED FUNCTION & PERFORMANCE ENHANCEMENTS

BALANCED SYSTEMS

- MANY OPTIONS FOR MATCHING CONFIGURATION TO WORKLOAD
- EXPLOITATION OF LARGE STORAGE CONFIGURATIONS
- INCREASED I/O CAPACITY TO MEET MIPS GROWTH

<21> CMG 87 EIC 12/87

A Computer and Communication Network Performance Analysis Primer, Prentice Hall, Englewood Cliffs, NJ, 1985; revised, 1987

B.Stuck and E.Arthurs
Probe Research, Inc. and Bell Communications Research, Inc.

CMG'87 New Books Section, Paper Reference # 21
New Books Session, Tuesday, 8 December 1987

Introduction

A significantly revised edition of a book entitled **A Computer and Communication Network Performance Analysis Primer** which first appear in 1985 has been issued. The latest edition involves over 25% new material, as well as significant revisions to over 50% of the old material, resulting in a substantially new book that has benefitted from a wide variety of constructive criticism in the marketplace.

The new edition concentrates much more on the commercial marketplace (read IBM), and on networking issues, than the first edition. The exposition was felt to be overly terse with respect to a number of case studies, and this issue has been addressed. The goal has been to make the concepts accessible to the widest possible audience: the breadth and depth have been increased. The intent is that the new edition should prove to be a very valuable reference book for professionals as well as an excellent introductory text for neophytes. Time will tell.

The book is organized in a teach by example method, starting from specific case studies and moving toward general principles. Typically an actual situation is described, to motivate the material; the next step is to quantify what performance means for this situation, what are the inputs and what are the outputs; next, a formula is presented (with a reference in the open technical literature to where it came from) that relates inputs, outputs, and performance measures, with no time spent on derivation of the formula (a very limited number of derivations are presented for the few results that are felt to be critical); finally, a variety of numbers are substituted into the formula, to gain insight into what parameter changes are most significant on performance and what are not, in order to gain qualitative insight into performance tradeoffs by means of quantitative numerical sensitivity studies.

Over two hundred exercises, over one hundred and fifty figures, over one hundred fifty references to the open technical literature, spread over thirteen chapters spread and six hundred pages (plus an answer book for the exercises for instructional use) make this book a valuable addition to the library of any analyst.

The exercises are an integral part of this material: working the exercises stresses the understanding of the concepts in new directions, and reinforces what is important and what is not.

Two problems are encountered in this material: first, the actual applications (taken from real world computer communication systems) may boggle the purist in complexity, while being much much too simplified for the practitioner. Second, the large number of equations and formulae suggest to the practioner that this treatise is too mathematical; in fact, there is virtually no mathematics here, only substitution of numbers into formulae, requiring addition and subtraction and multiplication and addition, all of which may alienate the purist.

Section A. Mean Value Analysis

A significant amount of new material has been included in this section. The discussion begins with static or deterministic scheduling, such as in a batch environment, with all the work present at an initial time, and the goal is to minimize the time required to execute all the work. A uniform representation of these problems, based on specifying the steps required for each job, the resources required for each step, and the time per step, fixes all the ideas. This representation, based on a standard table or tableaux of steps, resources, and duration, is used throughout the entire text; the earlier edition was criticized, and rightfully so, for not being consistent in this regard.

This analysis is extended by allowing more and more jobs to be present at an initial time. It then becomes meaningful to deal with the ratio of the total number of jobs executed in a given time, or a rate or throughput. No stochastic processes, no queueing theory, nothing but workflow concepts, fix all these ideas. Over one half of the book is devoted to this topic: how to apply it, where it is strong and weak, and example after example to reinforce these points. Other books typically give this roughly ten pages, dismissing it as intellectually trivial. Here we argue that this material is the most important part of computer communication system performance analysis, and should be emphasized in the extreme, not dismissed in passing.

For those that want more, we pose this analysis as a linear programming problem, with the possibility for bounding performance using widely available numerical LP packages. The advantage of this approach is that it is hierarchical: as more constraints are included, as the model becomes more complex and realistic, this phenomena can be quantified and included, and the performance bounds become correspondingly tighter. Nonetheless, the bounds are useful at any stage of analysis, with the caveat that the more simplistic the model, the looser the bounds.

Major new examples developed here include online lottery transaction processing (with a performance goal of processing over 2,000 transactions/second with an end to end response time (measured from the time the wager is entered at a terminal until the terminal finishes printing a ticket for that wager), and a voice storage system analysis for a telephone common carrier.

A great deal of emphasis is placed on parallel processing: what performance is achievable for a given workload and schedule (how can algorithms and data structures help?). This is unique for an introductory text.

The section on data analysis or model validation is well known to statisticians, but is rarely found in books on computer communications performance analysis. This is surprising, because the crux of this type of analysis is to use computers and graphics to generate plots and pictures that suggest what models are most appropriate for the data, and how good a model fits actual data.

The entire section on mean value analysis forms a self contained one semester course on performance analysis.

Section B.Jackson Queueing Networks

Jackson queueing networks are motivated as a limiting case of mean value analysis: by making stronger assumptions about a model, correspondingly stronger statements can be made about performance tradeoffs.

New examples to be found in this section include a database management system hierarchical performance analysis, and a voice/data network design for a major commercial bank. The network design encompasses state of the art methodologies, and is unique.

The mean value analysis is carried out in parallel with the Jackson network analysis, in order to emphasize that the Jackson network analysis is a refinement that quite closely approximates the mean value analysis.

Section C.Single Serially Reusable Resource Scheduling

In many computer communication networks, under load one resource becomes a bottleneck, and virtually all performance tradeoffs revolve around how work is scheduled on the bottleneck.

This section proceeds hierarchically, starting with an introduction to the area, priority scheduling impact on performance, and local area networks.

The introductory section includes a discussion of link level communication protocols, including a delay analysis for HDLC. This is unique to an introductory treatment.

In addition, the introductory section includes a discussion of numerically approximating delay distributions, not simply averages or higher order moments, using a class of flexible, fast and quite accurate algorithms. This is rarely found in any book.

The priority scheduling section includes a discussion of non preemptive and preemptive resume scheduling, including the impact of correlation of tasks (a job consists of a set up phase, a work phase, and a clean up phase, and each is run with its own priority in this order) and burstiness (using a compound Poisson arrival process with multiple arrivals at the same time epoch of different urgencies and type of work). These results cannot be found anywhere else.

The concluding sections of priority scheduling deal with link level flow control and voice/data multiplexing. This analysis cannot be found anywhere else.

The section on local area networks include the most up to date analysis yet on Ethernet-like networks, token passing networks (rings and busses), and hybrid networks that behave like Ethernet under light load and like token passing under heavy load. Excellent texts are now appearing on these topics. The analysis of Ethernet appears nowhere else.

Section D.Special Topics

The final area deals with detailed case studies, including recent advances in performance analysis of switching systems (circuit switching, packet switching, Batcher/Banyan, and hybrid), a multivendor layered protocol performance analysis (including SNA, DECNet, TCP/IP, and ISO), a refinement of the voice/data network design case study described in the Jackson network section, a case study in operating system versus computer networking performance tradeoffs, and reduced instruction set computer (RISC) performance tradeoffs (processor, cache, main storage, bus performance, compiler, and networking capabilities).

Because these topics are of interest to specialists, they can be used as part of a seminar, omitted entirely, or touched upon in a variety of settings. One of the advantages of a book is that everything need not be used in a single class, but can be spread over different classes and different audiences.

A CLOSE LOOK AT MVS SYSTEMS:

MECHANISMS, PERFORMANCE AND SECURITY

by

Ronald Paans, IBM The Netherlands[1]

The book "A Close Look at MVS Systems: Mechanisms, Performance and Security" provides a functional description of many hardware and software components of MVS systems, and reflects an investigation of the performance and security aspects of these systems[2]. During the presentation at CMG, an overview of the areas investigated and the major conclusions will be presented. Emphasis will be on performance and monitoring aspects, showing that the behaviour of hardware, MVS system software and applications can exhaustively be observed by researchers.

1. INTRODUCTION

A synopsis is presented of the book "A Close Look at MVS Systems: Mechanisms, Performance and Security" by Ronald Paans, ISBN 0-444-70008-0, 572 pages, published by Elseviers Science Publishing Company, Inc., New York. The aim of the book is to indicate rational and practical means of improving the *performance* and *security* of mainframes running under control of IBM's *Multiple Virtual Storage (MVS)* operating system. This modern control program is sophisticated, complex and widely used: many tens of thousands of MVS systems have been installed worldwide. During the years to come, MVS is likely to remain one of the important control programs for commercial and scientific applications running on mainframes with the IBM System/370 architecture as implemented in IBM 43XX, 308X, 3090 and their plug-compatibles.

The primary goal of any computer system is to perform the functions requested by its users reliably, correctly, securely, effectively and efficiently. *Quality* should be the prime aim in this context. Computing-centre staff responsible for capacity-planning, tuning, maintenance, trouble-shooting, security and data integrity need detailed information about all activities within a system and their relation to the users. Without appropriate information, any form of prediction remains mere guesswork, while improving performance and security is awkward and sometimes turns out to be impossible.

The success of data processing and office automation is closely related to the acceptance of systems by their users; the quality perceived is not only gauged by response times and availability figures, but also includes *user-friendliness* of systems and applications. A critical view of the interactions among systems, applications and users suggests that the intermediate software is designed by technicians to be used solely by technicians, thus excluding many users from understanding anything. Command syntax and data-set names have to follow incomprehensible rules, while messages are cryptic due to quirks of design and implementation, so often causing feelings of discomfort and confusion to the users interacting with applications. Some designers persist in defending inconveniences; they hide behind incomplete specifications, refuse to fix the code and retrofit manuals, thus legalizing incorrect software.

Before a system's *performance* can really be improved, that is lowering its response and turn-around times and raising its availability and accessibility, one must first exactly know how the system's mechanisms behave and interact with one another. A prediction of the consequent gain can be no more reliable and credible than the information and knowledge the experimenter has been provided with. In practice, this experimenter is confronted with a severe lack of information:

- the processing power available for applications is not explicitly known, nor has the useful capacity of the CPUs or the I/O subsystems been stated in quantitative terms,
- there are no methods to determine properly the resource capacity wasted due to improper design, implementation or tuning of system and application software,
- most computing centres do not have appropriate monitoring aids to analyse the system's activities and to distinguish unambiguously between avoidable overhead and relevant, productive work.

It follows that the combined ignorance of system activities and the absence of yardsticks prevents one from optimizing the quality of the services delivered. There is a yawning abyss between most of the values currently recorded by engineers and the performance and security as perceived by users and management. The book attempts to bridge this abyss, however precariously. One of its primary aims is to explain the method of operation of many components in MVS systems and to make some previously unobservable activities of these systems observable and quantifiable to engineers by suggesting new monitoring techniques. Observation and modelling allow engineers to understand the underlying mechanisms, to collect pertinent information on their behaviour and, combining knowledge and data, to improve the quality by re-design, re-implementation and tuning.

Since the author is an engineer rather than a mathematician, the emphasis is on applied rather than theoretical knowledge. Mean-value analysis has been applied to create the models described, but no new analytical methods have been developed. Written from a *systems programmer's* point of view, the book is directed at systems and application programmers, performance analysts, security officers and computing-centre management. Of course, it also contains information relevant to all interested in performance and security aspects of mainframe systems under control of MVS. Each Chapter discusses a single topic and has been written as a relatively self-contained paper preceded by an abstract and terminated by the conclusions on the topic highlighted. The list of Chapters reads:

1 Introduction

PART I. THE UNDERLYING HARDWARE AND CONTROL SOFTWARE BASES

2 The Hardware Base (architecture and logical design)
3 The Control Program Base

PART II. MANAGING AND CONTROLLING MVS SYSTEMS

4 Performance and Service
5 EDP Performance Management
6 A Topology for Secure Systems
7 Secure User SuperVisor Calls

---

[1] IBM, Dept. International Business Services (IBS SC), Zoetermeer, The Netherlands. This paper reflects the personal opinion of the author, which is not necessarily shared by IBM.

[2] The research described has been performed at Delft University of Technology, The Netherlands, in close co-operation with Prof. G.L. Reijns and Prof.dr. I.S. Herschberg.

PART III. PERFORMANCE ANALYSIS AND MODELLING OF MVS SYSTEMS

8 Characterization of TSO Users' Behaviour
9 The Performance of Application Software
10 TSO: Performance of an interactive Facility
11 SuperVisor Calls and their Services
12 Performance Aspects of Access Control
13 The Control Program's Nucleus

PART IV. ANALYSIS AND MODELLING OF MVS HARDWARE

14 Throughput of Instruction and Execution Units
15 Performance Improvements by Firmware
16 System and Application Disk I/O
17 Simulation of a Disk Cache for User I/O

PART V. MONITORING IN MVS SYSTEMS

18 Monitoring and Recording Techniques
19 Monitoring by Event Tracing
20 Tracing Instruction By Instruction
21 Recording the CPU's Instruction Execution Times
22 The Artificial User as a Performance Observer
23 System Design Desiderata to Support Monitoring Tools
24 Epilogue

APPENDICES

A Evaluation Nets
B Sample of a Service Level Agreement
C System/370 XA Instructions
D Data Collected by MVS System Trace

Some of the investigations described in the book will briefly be summarized below, supplemented with some of the major conclusions.

2. PERFORMANCE AND SERVICE

The user's chief concern with a computer system is that it performs the requested functions reliably, correctly, securely and efficiently in a human-friendly way (and, preferably, at reasonable cost). This statement is a *contradictio in terminis*. For instance, if in a systems the level of reliability and security is raised, the level of efficiency will decrease and the cost will rise due to higher verification overhead. A balance should be struck between these factors so that the transactions are processed at an acceptable price and at the required speed with the requested quality. This is, as matter of fact, not an easy job in our complex computing-centre environment.

All activities in a system, explicitly or implicitly requested by users have one common aspect: they use *resources* such as CPUs, memories, disks, tapes drives, other devices, people, software, documentation and procedures. When considering a *computing centre* as a production unit the resources are the input: the sum of their individual costs shows the operational costs of the centre. The output of a production unit is usually expressed in goods or services; in a computing centre the output consists principally of *services* delivered to its users. Some of these services are: executing a batch job, processing an interactive transaction, making available compilers and other software to users, and performing data-entry and data-enquiry activities. Each of the services is associated with one or more applications or facilities using software and hardware resources.

A computing centre with its users is a complex of people and systems, controlled by rules and procedures. All activities should directly or indirectly be devoted to producing 'useful work', which may be updating data bases, obtaining specified records from data bases, executing commercial or scientific programs, etc. The notion 'useful work' must be defined in the objectives for the computing centre. In this context, one has to distinguish between *effectiveness* and *efficiency*. A background for these terms has been given by Norton and Rau in their "Guide to EDP performance management" [1]. The key-note question for effectiveness is:

"Are the requirements of the users fulfilled by the services delivered?"

The key-note question for efficiency is:

"Are the total costs of data processing as low as possible?"

There is an essential difference between these two notions, which may be formulated tersely by:

Effectiveness = Doing the right things Efficiency = Doing things rightly

Obviously, *effectiveness* has its due place in a computing centre: doing the wrong things efficiently will not supply any gain to the organization but is only wasting manpower and other resources. When one considers the computing centre as a black box, it uses resources as input and delivers services as output. Both the input and output should be evaluated for effectiveness and efficiency, resulting in the formulation of four queries. Each query is characteristic for a particular point of view and presents a theme for research, viz.:

A1. The effectiveness of the services (the service requirements):
does each service delivered fulfill the requirements of the users ?

A2. The efficiency of the services (the load of each service):
what are the resource loads of each individual service and which part of these loads is squandering of resources ?

B1. The effectiveness of the resources (the resource requirements):
are the resources capable of processing the total workload including momentary peak loads ?

B2. The efficiency of the resources (the resource balancing):
are the resources utilized optimally during processing of the workload including the instances with peak loads ?

In the book, examples are discussed to elucidate this classification and to show the relation between the queries and day to day practice. Answering the four queries gives some insight in the computing centre's activities: for instance, it reveals whether one performs useless work. In some organizations, work without any earthly use is processed very efficiently. To avoid this kind of squandering manpower and other resources, one has to evaluate periodically the effectiveness and the efficiency of the resources as well as the services. The queries A1 and B1 are worked out in more detail showing the user's point of view while interacting with the centre's services. This discussion results in defining a set of performance indicators which make the user's satisfaction visible. Bounds for these indicator's values should be included in the centre's *Service Level Agreement (SLA)*. The latter document acts as a written warrant for a certain level of comfort during the human-computer interaction. In the author's opinion, the requirements of the users, dictated by the principal objectives of their organizations, are the *pivot* in the process of defining services and SLAs. As matter of fact, the user is the one who really works with the services and, after all, he supplies the money to run the entire computing centre. Especially in a service-bureau environment, neglecting the user's requirements will not only result in unsatisfied customers but also in loss of these customers, obviously implying loss of revenue. In this context, computing-centre management is encouraged to pay attention to the performance indicators usability, availability, accessibility, number of service interrupts, and timeliness, in order to prevent their shop from being closed.

3. EDP-PERFORMANCE MANAGEMENT

The performance of computer systems must not only be observed and recorded, but must also be managed and controlled in order to process the services in an effective and efficient way. The aims of *performance management* are (a) optimizing the quality of the services delivered to the users and (b) delivering these services with a minimal load on the hardware and software resources. In the book, performance management is broken down into eleven distinct tasks representing eleven performance functions:

1. definition of SLA,
2. verification of compliance with SLA targets,
3. capacity planning,
4. elimination of bottle-necks,
5. tuning networks,
6. maintenance of disk space,
7. tuning applications,
8. tuning services and service environments,
9. tuning the system's kernel,
10. trouble shooting and keeping the system alive,
11. solving user problems.

The tasks 1 to 3 are related to capacity planning and the quality of the services as specified in the SLA; this is chiefly long-term work. The tasks 4 to 9 embrace maintenance, tuning and elimination of bottle-necks, and deals with a planning on a weekly or monthly scale, while the tasks 10 and 11 consist of ad hoc operations in order to keep the system alive and to solve user problems. As shown for a hypothetical computing centre with a compartmentalized organization, each task requires appropriate tools and skilled personnel, and is assigned to a department with a first-line responsibility. In addition, a formal structure is defined for second- and third-line support. This scheme is supplemented by an overview of the tools required to perform each of these tasks.

The majority of personnel in a computing centre are greatly in favour of delivering high-quality services. However, they are often frustrated by a lack of manpower and funds, and hence fail to meet quality targets. Obviously, no organization will make unlimited manpower and funds available to performance evaluation and improvement, since that is only a one of the many aspects of data processing. But, in sad practice, many centres are guilty of false economies and do not actively support performance management which they regard as avoidable overhead. As a consequence, effective performance management in such a centre is doomed to failure: low quality of the services and complaining users are the predictable outcome.

4. CHARACTERIZATION OF TSO USERS' BEHAVIOUR

The input of interactive data-processing facilities are the transactions generated by users; the messages displayed at the users' terminals constitute the output of these facilities. In fact, a user may be characterized by his rates of generating transactions and of requesting messages. Using observational data collected during three years in a service-bureau environment, the significant sensitivity of the user to system responses is derived by correlation and regression analysis, expressing the transaction rate and the message rate of each user in function of the average response time. It is shown that an increase in response times leads to an *over-proportional decrease* in the users' transaction rate which may, under specified conditions, be translated into an increase in the costs of programming projects.

Regression analysis is applied to derive an equation for the effect of the average response time, *Tresp*, on the user's think time, *Tthink*, and so on his transaction rate, λi. Assuming a linear relationship between these two time-related variables, with a given user 'head-scratch' time, $Ti\emptyset$, and a constant *Ki* representing its slope, one has

$$Tthink = Ti\emptyset + Ki * Tresp$$

This equation represents a straight line in the (*Tresp*,*Tthink*)-plane. Regression analysis (least-square fits on the perpendicular distance between the straight line and each pair) has been applied to the observational data, showing for users interacting with IBM's Interactive System Productivity Facility (ISPF)

$$Tthink = 11.3 + 2.1 * Tresp$$

This equation states that the population is affected significantly by the response times: ***an increase of 1 s for Tresp results in an increase of 2.1 s of an ISPF-user's think time.*** The transaction rate of a user is

$$\lambda i = \frac{1}{Tthink + Tresp}$$

The elimination of *Tthink* results in

$$\lambda i = \frac{1}{Ti\emptyset + (Ki + 1) * Tresp}$$

or, with the values for $Ti\emptyset$ and *Ki* as found via regression analysis

$$\lambda i = \frac{1}{11.3 + 3.1 * Tresp}$$

Due to the limited range of observations, however, this equation should not be used beyond the range 0.50-3.00 s in *Tresp*. Thadhani [6,7] has derived a similar equation for internal host-response times in the range 0.25-3.00 s, though he found different values for the constants $Ti\emptyset$ and *Ki*. The observations for two object groups demonstrate that these constants characterize the users' behaviour and therefore depend strongly on the group observed, their experience and their environment (e.g., the facilities supporting them). This means that the 'constants' $Ti\emptyset$ and *Ki* have only a ***limited significance***: they are object-group dependent and should be computed anew for each specific user population.

5. THE PERFORMANCE OF APPLICATION SOFTWARE

A critical evaluation of application programs indicates that many of them work far from optimally. Current development strategies often fail to provide applications operating human-friendly, effectively and efficiently. Some underlying causes are failing management of the programs' development stage, higher-level languages not leading, by default, to efficient programs, and an unmanageable complexity of hardware and software combined. In addition, applications are often inefficiently installed and their parameters are adjusted far from optimal. These pessimistic statements about applications are elucidated by examples.

Properly defining data-processing functions and converting these functions into programs is widely granted to be difficult. Most people do not have an analytical mind, cannot grasp a sequence of data-processing activities, and fail in correctly, unambiguously defining data items and data manipulations required. Of course, the weak definition capabilities of human beings are not only reflected in applications, but also in service programs and operating systems. However, the latter evolve into effective software somewhere in the future because they are improved from time to time by new versions and releases. This evolution may be characterized as:

> The adversary process of trial and error often leads, after a number of versions, releases and modifications, to effective and efficient programs.

For instance, IBM's Structured Programming Facility (SPF) is better attuned to users than the conventional set of Time Sharing Option (TSO) commands which it replaces, and is undoubtedly more human-friendly. Nevertheless, SPF showed still many inconveniences in human-machine interaction; currently some of these inconveniences have been

removed by SPF's successor Interactive System Productivity Facility (ISPF), especially by version 2 of ISPF. Following this line, one may expect that a future version *n* with *n*>2 will contain yet fewer inconveniences and will be yet more human-friendly. The only disadvantage of such a growth process is that the current software is never optimal and the user has to be equipped with sufficient patience.

An ambiguous, incorrect and incomplete definition of functions is not the only weakness of current data processing: in the stage of code generation and implementation, many inconveniences are introduced too. While ineffectiveness is principally due to definition errors, inefficiency is caused by improper code generation, implementation and parameter setting. Currently, optimally functioning software still requires human inventiveness, experience and skills. As shown in the book, performance analysts consulting during the development stage, observing existing programs and recommending improvements may still greatly contribute to the efficient use of a computing centre's resources. Tools and manpower should be made available for these purposes.

6. TSO: PERFORMANCE OF AN INTERACTIVE FACILITY

While in early days of electronic data processing the workload principally consisted of batch jobs, the current trend is towards executing the major part of the workload in the foreground, thereby establishing direct interaction between the computer and its users. The success of data processing and office automation is closely related to the acceptance of systems by their users; the quality perceived is not only gauged by response times and availability figures, but also includes user-friendliness of systems and applications. A critical view of the interactions among systems, applications and users suggests that the intermediate software often is designed by technicians to be used solely by technicians, thus excluding many users from understanding anything. Attention is focussed on the performance of Time Sharing Option (TSO) discussing resource loads and user-friendliness of some of its components. The major objects under observation are editors, data management, foreground processing of data, and background processing controlled by foreground commands.

7. PERFORMANCE ASPECTS OF MVS ACCESS CONTROL

A quantification of the invocation rates of an access-control package is a prerequisite for an evaluation of the resource loads occasioned by such a package. During the investigation described, the workload of a computing centre with four MVS systems has been characterized, formulas have been derived describing the relation between workload and access-control activities, and the invocation rate of access control has been quantified. The method developed is principally based on counting SMF records and is easy to use.

While in the versions up to 1.4 of Resource Access Control Facility (RACF) a system's owner was able to control RACF's resource loads by implementing exit routines, these loads may now be controlled effectively by Global Access Checking (GAC) and generic profiles since RACF 1.5. The effectiveness of these new mechanisms has further been enhanced by the RACF versions 1.6 and 1.7. Contrasted with RACF 1.4 in native mode (without performance improving exit routines), the significant load-reducing impact of Global Access Checking and generic profiles will be demonstrated.

8. THE CONTROL PROGRAM'S NUCLEUS

The activities of an operating system are shown to be observable by recording timed events at the level of Dispatcher activities and interrupts. A dedicated monitor, termed SYSTRAC, has been built to copy descriptions of these events, the so-called Trace Table Entries (TTEs), from the MVS System Trace Table. Analysing the stream of related pairs of events, termed bigrams, reveals how an operating system behaves and which parts of the resources are used by the major routines of the operating system's nucleus. Such an analysis provides a breakdown of the factor *overhead* by assigning fractions of the resource loads to particular user-initiated activities. When properly separating what is due to the users' activities and what is purely for the operating system, the latter becomes quantifiable as a minor contribution. The major contribution to the resource loads is work done on behalf of users and therefore should not count as overhead. For instance, initiating I/O for catalog and directory access is often wrongly included in the overhead; one then forgets that both catalog and directory have to be scanned to find information requested, explicitly or implicitly, by a user. Similarly, many other operating-system activities also are indispensable to run and support user software. In the book, the behaviour of some major software components is highlighted, viz. I/O Supervisor, interrupt handlers and Dispatcher.

The first component discussed is *I/O Supervisor (IOS)* of MVS systems running in 370-mode (non-XA), which contains two major groups of routines:

1. I/O Supervisor Front End with its IOS Drivers such as the EXECUTE CHANNEL PROGRAM (EXCP) Driver, initiating I/O activities so as specified by I/O-requesting programs,
2. I/O Supervisor Back End acting as Second Level I/O Interrupt Handler which informs I/O-requesting programs about I/O completion.

In this scheme, the First Level I/O Interrupt Handler plays only the minor role of saving the status of the task interrupted and branching to I/O Supervisor Back End for further handling an interrupt. By timestamping the invocation of an I/O Supervisor Driver, the moment of issuing a START I/O (SIO) instruction, and the moment of returning control to the I/O requestor, the CPU loads by I/O Supervisor Front End are calculated. In addition, timestamping the penetration of an interrupt as well as recording the next event reveals the CPU loads by I/O Supervisor Back End.

The second component highlighted is the *Program Check Interrupt Handler*, processing interrupts due to program errors and page faults. The investigation deals only with page-fault-handling activities; page faults occur frequently in MVS systems and are inherent in managing virtual storage by demand paging and page stealing.

The third and last functional component discussed is the *Dispatcher* responsible for passing CPU control to tasks under control of a Service Request Block (SRB) or a Task Control Block (TCB). When lacking dispatchable work, control is passed to the Wait Task causing the CPU to revert to the wait state. The latter activity - dispatching the Wait Task - is one of the few system activities to be regarded as *pure overhead*. However, this does not affect a system's performance because any and all user tasks are then waiting for events to occur such as I/O completion or timer expiration.

Observations in a number of systems over a period of four years have revealed that the CPU loads per activation of many operating system's routines are functions not sensitive to the momentary workload. For instance, the time required after a page fault to build a channel program for a single page-in transfer is relatively stable and is independent of the page-fault rate. Conversely, new versions, releases and modifications of the system turn out to affect these CPU loads; these impacts occasionally are drastic. On the one hand, this is a disadvantage of the method of analysis adopted because one has to requantify the resource loads for every new version, release and modification level of the system. On the other hand, it is also an advantage, because now one can also quantify the impact of any modification and is thus capable of determining the amount of CPU time saved (or lost) by these modifications.

9. THROUGHPUT OF INSTRUCTION/EXECUTION UNIT

One of the first parameters to characterize the performance of a computer system was its MIPS (Million Instructions Per Second) rate. In modern machines with microcode, prefetch mechanisms, pipelines and cache memories this term has been rendered obsolete and is no longer specified by the mainframe manufacturers. Admit-

tedly, in modern hardware, a single MIPS rate is unsuitable for characterizing CPU-power, the reasons being, amongst other things,

- it depends on the instruction type and the length and relative position of the instruction operands: among others operand overlap and byte access instead of full data-path access have a major influence on the MIPS rate for some instructions,
- it depends on the engineering level of the machine and on the microcode: any level modification can change the CPU time required to execute any and all instructions, and influences the total MIPS rate,
- it depends on the hit rate of the CPU's cache memory: hence, it is strongly influenced by the program behaviour,
- it depends on the instruction sequence: unexpected branches may destroy the contents of prefetch buffers and the pipeline.

Both the use of instructions and their *Instruction Execution Times (IETs)* in function of operands and environments have been observed in computers with the System/370 architecture. In the book, observational results are presented demonstrating, among other things, that IETs are suitable for observing and analysing the behaviour of hardware components in a CPU, and may be used to quantify the impact of performance improving mechanisms such as pipelines and branch-prediction logic.

Tables are presented with detailed information about the behaviour of four groups of programs and about the timing characteristics of seven types of CPUs. These Tables may be used by researchers to model and to simulate CPUs, multi-processor systems and cache memories. In academical environments (but not only there), researchers are often confronted by a severe lack of technical information due to the absence of large systems for exploratory purposes and suitable monitors. Moreover, they are often handicapped by insufficient experience in creating test environments and running monitor programs in order to collect relevant information. Hence, they all too often take refuge in using obsolete data such as Nessett's location-counter distributions (recorded in 1966 [2]) or Shustek's instruction-execution rates (recorded in 1978 [5]). With the Tables presented, researchers working at their desk with analytical and simulation models have up-to-date information at their disposal. The book also shows them ways to construct monitors to obtain the information required, even if their hardware is dissimilar to that dealt with here.

Another aim of this investigation was to prove that just two locally developed software monitors - PPTRAC and IETMON - are sufficient to obtain all relevant information on program behaviour and CPU characteristics, incidentally implying the obsolescence of hardware monitors. All relevant data, even at the machine instruction and hardware level, has become available to *software* monitors in modern MVS systems.

10. PERFORMANCE IMPROVEMENTS BY FIRMWARE

Computer manufacturers from time to time raise the effective processing power of their products, which power may be expressed in terms of work per unit of time as perceived by those in charge of the system. Two ways for increasing that power are open: building faster systems and improving the performance of existing systems. History shows that most manufacturers use both methods alternately. Due to compatibility requirements, the creation of faster systems usually will not affect the system's basic architecture but is limited to speeding up the hardware, e.g. by progressive integration of circuits. The second alternative, performance improvement, is most likely to be realized by software modifications in the operating system (OS) and in applications in order to process a given load more efficiently. When evaluating the performance of a system, one usually starts by constructing an ordered list of the routines with the highest resource loads, say a top-five list. The ordering is by decreasing values of the multiplication
*<activation rate> * <resource load per activation>*.
The rationale of this ordering is corroborated by Kendall [3], who states that only 1.5% of the application programs in a computing centre account for at least 50% of the total CPU load. This is not only true for applications, but also for the OS routines involved. Hence, the top-five OS routines are naturally most eligible for improvement, either by rewriting their software or by (partly) converting them into firmware. In this context, firmware is understood to be microcode, possibly assisted by associated hardware. Once it has been decided to incorporate frequently used OS functions into firmware, new machine instructions arise. From a programmer's point of view, new firmware is equivalent to additional hardware: in essence, a *new machine* is created with a *new architecture*, containing the new instructions serving (partially) to replace former software in addition to the original instruction set.

To quote but one example of such an evolution, in 1977, simultaneously with the announcement of the IBM 3033 CPUs, a new release of MVS was introduced: MVS/System Extension (MVS/SE). It showed a considerable performance improvement in comparison to earlier (pre-SE) releases. Many parts of its software routines had been rewritten and a set of new firmware-implemented instructions had been created, many of them completely replacing former software routines. Data-processing magazines (though non-certified sources) claimed a throughput improvement of some 14% in comparison with pre-SE systems. In order to make this gain available to computing centres with System/370 CPUs already installed, the System/370 Extended Facility was introduced at the same time, which upgrades any System/370 158 or 168 CPU by creating *fourteen new instructions* activating the new firmware and by adding several new hardware features. This facility supported three ends: improved system integrity, enhanced performance, and closing gaps in the instruction set.

By hindsight, the book attempts to reconstruct the research performed by the designers of System/370 Extended Facility, and quantifies the CPU load of the fourteen functions both in the pre-SE software implementation and with the new firmware. As will be shown, especially with the modern IBM 308X hardware, the firmware approach significantly reduces the CPU load of OS routines often invoked: a saving of some 75% of a routine's CPU load can be achieved. The data presented has academical value only since the new versions of MVS cannot run without this additional firmware. However, the key-note trade-off *'firmware versus software'* for frequently used routines remains relevant to computer architects, both for present and for future systems. This major trade-off is therefore addressed, attempting to balance the pros and cons of the competing approaches in the light of flexibility versus effectiveness.

11. SYSTEM AND APPLICATION DISK I/O

The architecture of the System/370 disk-I/O subsystem for non-XA systems is a logical outcome of decisions made during the design of the System/360 architecture in the late fifties and early sixties. In those days, logic circuits were relatively expensive. Principally due to economic reasons, the architects concentrated the intelligence in the channels located near the Central Processor Unit (CPU) and main memory. In this context, a channel acts as an I/O processor executing channel programs constructed by programmers or by access methods. Such a channel program is a chain of Channel Command Words (CCWs) each representing an instruction for a channel or a device.

Disk I/O in an MVS system is initiated by both system and application software. Their I/O requests have one aspect in common: all are issued on behalf of users or their batch jobs. However, these users specify only a part of the stream of I/O requests *explicitly* by, e.g., issuing an EXecute Channel Program SuperVisor Call (EXCP SVC) for reading or writing data, or a LOCATE SVC for retrieving catalog data. For these explicit I/O requests, identifying the requesting user is easy to do. The remaining part of the stream of I/O requests is an *implicit* result of the combined activities of all users; this comprises, e.g., paging and swapping necessary to support the users' virtual address spaces. This implicit part of the stream of I/O requests is obviously caused by users but it is awkward to identify for each I/O activity the user requesting it.

To raise the level of confusion, the designers of the standard MVS tools System Management Facilities (SMF) and Resource Measurement Facility (RMF) have introduced a distinction between direct and indirect work which does not coincide with the distinction between explicit and implicit work. SMF and RMF classify all work executed in problem-program mode and all SVCs issued in that mode to be ***direct work*** and the remaining work to be ***indirect work***. By definition, all I/O requests issued when running in supervisor mode are considered as indirect activities and, consequently, are accumulated in the opaque set of ***system overhead***. Observations in production environments often show that some 50% of the START I/O (SIO) instructions are accounted for as overhead, while this fraction is sometimes even 70% in a TSO environment. These observations demonstrate the ambiguity introduced by the distinction between direct and indirect work. Obviously, modelling the disk-I/O subsystem is bound to fail when one attempts to found such a model on observational data collected by SMF and RMF. One of the principal objectives to develop locally the SYSTRAC software monitor was to obtain, by exploiting MVS System Trace, pertinent figures about the I/O load of individual user tasks. After implementation of SYSTRAC, this monitor demonstrated to be suitable to identify the cause of each I/O request and to pin-point the task responsible for that request, except for the fraction of paging I/O related to page stealing. Using this detailed information, one may construct an accurate model for disk I/O describing channel-busy and disk-access times.

Within MVS systems, one may distinguish among several types of disk-I/O - termed the I/O classes -, each with its typical behaviour. For instance, if a TSO user requests to access a data set specified by name, he has first to create a logical path to this data set by allocating it through the DYNALLOC SVC. The underlying SVC routine has to access the catalogs to find the name of the volume, then the Volume Table Of Contents (VTOC) to obtain the data set's attributes, and can only thereafter build the control blocks establishing the requested allocation. These I/O requests for system-data sets such as catalogs and VTOCs are issued by the DYNALLOC SVC routine, running in supervisor mode while processing the user's requests (and, hence, are accounted for as indirect work).

In the book, a distinction is made among twelve classes of I/O, most of them representing indirect I/O. Moreover, their attributes are discussed and the relation is demonstrated between operating-system functions requested by users and these I/O classes. These functional descriptions apply to both non-XA and MVS/XA systems.

12. MONITORING BY EVENT TRACING

The research presented in the book covers many in-depth investigations on the behaviour of system and application software in MVS systems. Evaluation of the important mechanisms in the operating system's kernel, such as the Dispatcher and the interrupt handlers, requires a monitor providing pertinent information about these mechanisms' behaviour and activities. At the start of the research, an evaluation showed that none of the commercially available monitors was suitable for supplying this information. For instance, the rejection of hardware monitors was dictated by the lack of appropriate wire-wrap pens on the back-planes of modern CPUs which formerly served to connect the monitors' probes. In addition, Generalized Trace Facility (GTF) had to be rejected due to its interference with the observation object, making CPU-time calculations meaningless. It was hence decided to develop a new monitor - termed SYSTRAC - based on MVS System Trace, in order to obtain pertinent information at a level of detail and accuracy not to be found in any of the commercially available tools. MVS System Trace collects data about some types of events associated with system activities (starting a device, handling an interrupt, dispatching a task, etc.), constructs 32-byte records and stores these records into the System Trace Table as Trace Table Entries (TTEs). The software monitor SYSTRAC samples this table and copies, in addition, some relevant fields of the MVS control blocks. This data-collection operation is exercised without interference with the object programs' activities. The data collected is recorded into data sets making it so available for further analysis and amenable to simulations. The types of TTEs recorded are:

- Start I/O operation
- external interrupt (timer, signal of other CPU, etc.)
- SuperVisor Call (SVC) interrupt
- program-check interrupt (page fault, etc.)
- dispatch of work under control of a Service Request Block (SRB)
- I/O interrupt
- redispatch of work under control of an SRB
- dispatch of work under control of a Task Control Block (TCB)
- return from SVC processing
- Program Call (invoking a program in a secondary address space)
- Program Transfer (returning to the primary address space)
- set Secondary Address Space Register

A set of programs has been developed to process and analyse this data. The method of operation, internals of the SYSTRAC monitor and sample output are presented in the book. The software monitoring technique of event tracing, as implemented in SYSTRAC, turned out to be capable of supplying ***all*** data relevant to performance research. However, there was one practical problem: upgrading SYSTRAC to keep pace with new releases of MVS turned out to be a severe bottle-neck, chiefly due to lack of skilled man-power at Delft University of Technology, where this research has been done. Within a few years time, SYSTRAC had been upgraded for the successive moves from MVS/non-SE to MVS with System Extensions version 1 (SE-1), SE-2, MVS with System Products version 1.1.1 (SP-1.1.1), SP-1.3.1 and SP-1.3.2. At that moment, just before the introduction of 31-bit addressing of MVS/XA, further upgrading has been renounced. Although the current implementation is still dedicated to non-XA, the monitoring method also applies to XA systems.

13. TRACING INSTRUCTION BY INSTRUCTION

The IBM System/370 architecture does not supply hardware facilities for ***instruction-by-instruction*** tracing. Some computer architectures, such as the PDP-11 and the earlier IBM Series 7000, contain dedicated hardware for tracing user software at the level of individual instructions, so supplying opcodes, addresses referred to, operand lengths and operand values. This data is useful for:

- debugging purposes,
- obtaining insight into the program's behaviour,
- optimization of user programs and applications,
- evaluation, by simulation or analytical modelling, of cache-memory performance and CPU throughput in existing and future systems,
- design of systems.

Because of the absence of hardware facilities in the Systems/370 architecture, a software monitor has been designed to obtain detailed information on the behaviour of programs in MVS systems. It acts as an interpreter or a virtual machine, traces software and records the opcodes executed, the addresses referred to, etc. Its principal application is to collect data suitable for evaluating the performance and throughput of the CPU and its components. This monitor, somewhat confusingly termed Problem Program Tracer (PPTRAC), is currently capable of tracing both problem-program-mode and supervisor-mode single-tasking software running with any key-in-PSW. With PPTRAC, routines of the MVS operating system have been observed such as the Dispatcher, Interrupt Handlers and SVC routines. The data collected is stored in a data set with a compact structure. In the scope of the research described in the book, several trace data sets has been created spanning a variety of programmers and languages. Five data-analysis programs (postprocessors) have been written to process the data collected; their functions are

1. simulating a single-CPU cache memory supporting the store-through and store-in-cache algorithms in direct-mapping and set-associative cache organizations,

2. reporting the distribution of the distances between instruction-fetch addresses and between storage-operand addresses,
3. reporting the number of instructions and the distributions of operand lengths arranged by instruction format and data type,
4. observing the amount of data stored in a cache in function of the time starting at a context switch (which, de facto, has the same effect as a reset of the cache),
5. reporting the use of instructions ordered by opcode, the number of first and second operand fetches, and the number of result-store operations.

The method of operation of PPTRAC and sample output is presented in the book. In this context, we introduce the notions *parent* for the tracing program (PPTRAC) and *child* for the traced program (a user program or a part of the operating system). PPTRAC fetches, just as the CPU's Instruction Unit, child instructions one by one and inspects their opcodes. After some verifications, PPTRAC let the CPU execute each child instruction or, in case of e.g. a branch, interprets it. Many precautions have to be taken in order neither to loose control nor addressability. Not only branches, but also SuperVisor Calls (SVCs) may disrupt the sequential instruction stream. Hence, SVCs such as LINK, PASS CONTROL (XCTL), OPEN, SPECIFY PROGRAM INTERRUPTION EXIT (SPIE) and END OF VOLUME (EOV) need special processing. Another problem is caused by the practice of System/370 to compute addresses as sum of a displacement and the contents of one or two registers. Since the child is allowed to use *all* registers, the parent has, during execution of a child instruction, *no* register at its disposal and, as a logical consequence, cannot address its own storage locations. Some software tricks had to be implemented in order to establish addressability for the parent and to guarantee control.

After successful completion of the problem-program-mode part of PPTRAC in 1982, it was decided to extend the monitor for single-tasking supervisor-mode software with a key-in-PSW less than eight (the system keys). This extension was required to trace parts of the operating system such as the Dispatcher, Interrupt Handlers, and SVC routines; it implied roughly a triplication of the monitor's size and caused many complications. For instance, parts of the operating system run on a CPU disabled for interrupts. Hence, during interpretation, PPTRAC also has to use this disabled CPU and therefore can neither issue SVCs nor start I/O to store the data collected on disk. Even after solving all problems, we never have dared tracing supervisor-mode software during production hours so as to avoid the risk of interfering deleteriously with other users.

14. RECORDING THE CPU'S INSTRUCTION EXECUTION TIMES

A Central Processing Unit (CPU) has only a single primary purpose: it has to process operands at programs' requests. Each unit of the CPU's processing effort is explicitly specified by an *instruction*, consisting of an operation code (opcode) and, optionally, a specification of operand(s), all located in virtual storage. A hardware pointer - termed the Program Counter - is located in the *instruction-address* field of a CPU's Program Status Word (PSW) and designates the next instruction to be executed. This pointer is *de facto* controlled by the program currently in execution; its contents are incremented during execution of a non-branching instruction or are, when branching, replaced by a new address. The opcode of the current instruction is interpreted by the CPU and the operation specified is executed by hardware, often in conjunction with microcode. One to three instructions may be prefetched and processed simultaneously with the current instruction; sophisticated mechanisms attempt to predict the outcomes of branch instructions prior to execution so as to minimize disruption of preprocessing by unexpected branches. The CPU time required depends on the complexity of the function requested and on the presence of all relevant data within the highest-speed circuits of the CPU (registers, pipeline and cache). All data not found there has to be accessed from the slower circuits of main memory. The factors affecting the CPU time are:

- the type of the instruction (e.g. LOAD, SUBTRACT DECIMAL),
- the length of the operands (e.g. 16 bytes for an ADD DECIMAL),
- the distance between the operands (e.g. in the special case of a MOVE CHARACTER, transferring 255 bytes over a distance of 1 byte causes a decelerating destructive overlap),
- the environment of the instruction: e.g. unexpected branches disturb the operation of prefetching and preprocessing mechanisms, decrease the degree of parallelism, and raise the instruction's execution time.

Usually, a CPU's capacity is expressed as a MIPS (Million Instructions Per Second) rate. This number is the reciprocal of the average Instruction Execution Time (IET, in microseconds), which equals the average CPU time per instruction. One should be aware that the average IET depends strongly on the software, specifically on the mix of instructions, their operands and the sequence of instructions. Some of the investigations described in the book require knowledge of IETs for individual instructions in function of their operands. For this purpose, a software monitor - termed *IETMON* - has locally been developed executing instructions in simple loops and recording the CPU time used. Combining observations for over eighty instruction sequences and for some 3000 combinations of their operands, the IETs are deduced. This data is included in the book in the form of tables. In spite of this monitoring technique's simplicity, the output of this monitor has proved to be highly reliable.

With the IETMON monitor, one is able to evaluate accurately the impact of hardware changes in a system such as the replacement or upgrade of a CPU. For instance, IETMON has been used to quantify the improvements caused by the upgrade of an IBM 3081-K32 to a 3081-KX32. At a first glance of these reports, one is struck by the reduction in the CPU's cycle time from 26 ns to 24.5 ns, which change was reflected in all those IETs being a multiple of the cycle time. In addition, such reports disclose many other speed-improving measures introduced by hardware and microcode upgrades.

15. THE ARTIFICIAL USER AS PERFORMANCE OBSERVER

From a user's point of view, both the host computers and the networks are black boxes. The user communicates with these boxes through his terminal and is *only* interested in the performance at his specific location. He does not know nor does he want to know how all components in the networks and the host systems operate and interact, neither is he interested in delay times per component nor in availability per network link. His interests in performance are limited to:

- availability,
- accessibility,
- number of service interrupts,
- responsiveness,

all as perceived at his terminal. These performance indicators are affected by the behaviour of a large number of components in the networks and hosts, and often cannot be recorded by any program running on a host. When one is interested in actual values and historical trends, the lack of intelligence in most terminals implies the need of some human beings recording by hand, through sampling, the performance indicators at their terminal and processing the observational results. This way of monitoring is obviously expensive in manpower. On closer examination, a *user* recording performance indicators may be considered a *generator* for transactions and a *recorder* for timestamps. As a generator, he is characterized by his rate, λi, of issuing transactions and by the type of these transactions. This generator does not work standalone, but is fed with messages created by the host; it should react to the contents of messages such as 'READY', 'CANCELLED', 'IKJ1234', 'TERMINAL LOCKED' and 'MAXIMUM NUMBER OF USERS ALREADY LOGGED ON, TRY LATER' by issuing appropriate transactions. This reaction mechanism con-

sists of a feedback from the message-receiving function (= terminal) to the transaction-generating function (= user). In addition, the user also acts as a recorder logging the timestamps of sending transactions and receiving the host's responses. Such a user may be replaced, with advantage, by a dedicated (micro)computer programmed to act as a message-controlled generator and as a recorder.

Based on this philosophy - replacing a user by a micro -, a number of monitoring aids have independently been developed by several universities and companies. Two implementations are discussed: the RESPMON monitor of Delft University of Technology and IBM's Automated Response Time System (ARTS). The architecture of this type of monitor - the ***artificial user*** - turns out to be simple-to-implement and, once implemented, easy-to-use and providing accurate figures about the performance as perceived by real users. During many years, the artificial users proved to be reliable and accurate observers, and, once implemented, are substantially cheaper than human beings. This type of monitor is very suitable for observing availability, accessibility, number of interrupts, and response times as perceived by the user at his terminal while he is interacting with the large black box containing the networks and hosts. In this context, the monitor provides the data required for defining Service Level Agreements, and also is used to verify compliance with the targets mentioned in these contracts between a computing centre and its users.

16. EPILOGUE

After writing all papers dedicated to performance, effective use and security of MVS systems, brought together in the current book, the author was confronted with the question "Is there still a future for large systems?". This question was inspired by the many similarities observed between the current evolution of mainframes and that of, e.g., species such as the dinosaur. The latter grew bigger and bigger while all their natural enemies disappeared or were, after a certain instant, too small to endanger these giants. And then, within a relatively short time, the entire population of dinosaurs dwindled. Are mainframes, also growing bigger and bigger, be doomed to the same destiny as the dinosaur?

On the one hand, data processing is being ***distributed***, rapidly moving parts from a few mainframes to large populations of stand-alone Personal Computers (PCs) and groups of PCs organized in Local Area Networks (LANs). Personal computing at one's desk provides many advantages to the endusers, has conquered a fast-growing market and has become very popular in many disciplines. However, there are still many problems to be solved before distributed processing reaches maturity: these problems lie, e.g., in the areas of security, data integrity, backups, maintenance of auxiliary storage, cataloging, performance and compatibility. On the other hand, many organizations are confronted by the necessity for ***large data bases*** with distributed data-entry and data-inquiry capabilities and with applications requiring large amounts of CPU and I/O power. These organizations still need mainframes and will at least continue to use these during the next decade. As a consequence, the mainframe market is still a growing market both in number of computers installed and in the CPU and I/O capacity of the individual systems.

In the author's opinion, we shall grow, during the years to come, to see groups of mutually connected mainframes, front-ended by many LANs and stand-alone PCs through their TP networks. Each organization will feel the need for central, large data bases parts of which are copied through the networks to PCs for local processing and are updated and maintained by PCs. As a consequence, the question "Is there still a future for large systems?" may be answered in the affirmative. At least during the next decade, systems programmers, application experts and, of course, people with network skills have not to fear unemployment.

17. REFERENCES

[1] Norton, D P and Rau, K G 'A guide to EDP performance management: system development, computer performance operations' QED Informations Sciences Wellesley Mass (1976) ISBN 0-89435-021-8

[2] Nessett, D M 'The effectiveness of cache memories in a multi-processor environment' The Australian Computer Journal Vol 7 No 1 (Mar 1975) pp 33-38

[3] Kendall, R C and Lamb, E C 'Management perspectives on programs, programming and productivity' Tutorial Programming Productivity: Issues for the Eighties (ed. Capers Jones) IEEE No EHO 186-7 pp 201-211

[4] Paans, R 'A Close Look at MVS Systems: Mechanisms, Performance and Security' Elseviers Science Publishing Company, New York (1986) ISBN 0-444-70008-0, 572 pages

[5] Shustek, L J 'Analysis and performance of computer instruction sets' Ph.D. Dissertation STAN-CS-78-658 Stanford University (1978)

[6] Thadhani, A J 'Interactive user productivity' IBM Systems Journal Vol 20 No 4 (1981) pp 407-423

[7] Thadhani, A J 'Factors affecting programmer productivity during application development' IBM Systems Journal Vol 23 No 1 (1984) pp 19-35

CICS - DESIGNING FOR PERFORMANCE

Eric Emanuel
Candle Corporation

ABSTRACT

"CICS - Designing for Performance" is a soon to be published book providing a unique approach to CICS Performance management. Based on a real world practical approach to include operating systems and other subsystems that CICS may interface with. The major emphasis is on total system performance and how CICS best fits into the overall scheme of things. An example of this scenario would be the design of a CICS application using MRO, ISC, DB2, DL1, as well as a combination of other system components. Performance analysts, applications programmers, and capacity planners are the primary audience. Valuable information is provided for data processing and information systems managers as well.

INTRODUCTION

CICS is about to enter its 20th year as an IBM product. CICS evolution over these years has kept pace with the phenomenal growth of transaction processing. Business requirements coupled with technological advances have transformed CICS from "cost saving" back office applications, such as order entry and inventory control to "revenue earning" consumer direct service applications, such as home banking, and point of sale electronic funds transfers. The high visibility of CICS to the consumer marketplace coupled with the growing size and complexities of CICS networks certainly brings the subject of successful CICS performance management into the limelight. Managing CICS to successfully achieve service level objectives with maximum availability and reliability is the key item to the success of an organization that depends upon CICS as a revenue generator.

The book discusses CICS industry trends, requirements, performance objectives, along with CICS system design and performance analysis techniques to help ensure that your organizational requirements and objectives are met. "CICS - Designing for Performance" consists of seven parts, with a total of 20 chapters. These seven parts are:

- Part I - CICS Overview and Evolution
- Part II - CICS Performance Objectives
- Part III - Performance Management Tools and Techniques
- Part IV - CICS Performance Analysis Techniques
- Part V - Improving CICS Performance
- Part VI - Designing High Performance Applications
- Part VII - Planning for Future Growth

This paper is organized to present an overview of selected parts of this book, and is organized as follows:

- Section 1 - CICS Overview and Evolution
- Section 2 - CICS Performance Analysis Techniques
- Section 3 - Planning for the Future Growth

SECTION 1 - CICS OVERVIEW AND EVOLUTION

The introduction of the IBM System 360 in the mid 1960's revolutionized the computer industry. This new hardware and its associated software products created new capabilities and opportunities that were previously unavailable to the business community. From the mid to late 60's, various attempts were made to produce a good general purpose transaction processing system. This was not an easy undertaking.

Three basic requirements had to be met. First of all, support of a varied terminal network including both local and remote terminals was a key requirement. Secondly, relative ease of implementing applications had to be considered. And thirdly, the expenditure of people, processing, and dollar resources had to be minimized. In 1968, IBM in cooperation with a major utility company developed CICS systems primarily for the utilities industry. The first CICS systems to be implemented were used primarily by customer service representatives in the utilities industry to provide quick turnaround time on customer inquiries regarding utility service and billing. CICS at that time was supported by the OS 360 operating system, BTAM was used for data communications with the terminal network, and ISAM and BDAM were the supported file access methods. CICS application programs at that were supported file access methods. CICS application programs at that time were written in Assembler language with macro level calls to CICS services.

In 1969, IBM projected the total number of CICS users worldwide would be about 50. Boosted by the application language support of COBOL and PL1 in 1970, the use of CICS started to grow at a phenomenal rate. CICS support of COBOL and PL1 significantly bolstered the availability of application programmers for the development of CICS applications. The phenomenal growth of CICS was accelerated greatly with IBM's introduction of 370 processors and the VS operating system. As virtual systems evolved, CICS kept pace with support of VSAM and VTAM (Figure 1 illustrates the phenomenal growth of CICS over the years).

The introduction of CICS Release 1.5 in 1980 with Multi-Region Operation (MRO support) and Intersystem Communication (ISC support) proved to be a major milestone, providing a "single system image" to the end user. The ability of CICS to run as multiple logically connected regions within the same or within

remote processors removed many of the growth constraints that previously inhibited CICS from supporting very large terminal networks.

Today there are approximately 30,000 CICS installations worldwide. This number is growing quickly. CICS has evolved from "back office" applications such as inventory control and order entry in the '70's to "front office" applications such as electronic mail and automatic teller machine banking in the early '80's to today's support of consumer "direct service" applications, such as home banking and point of sale funds transfers. The "back office" applications of the '70's and "front office" appls of the early '80's certainly provided cost savings benefits to those companies that have implemented these applications. The addition of today's direct service applications where the consumer is in fact the CICS end user provides these companies with some excellent revenue generating opportunities. CICS's excellent support of other products such as DB2 helps to verify CICS as IBM's strategic transaction processing product well into the 1990's.

SECTION 2 - CICS PERFORMANCE ANALYSIS TECHNIQUES

The bottom line performance of CICS is measured by two factors, response time and throughput. When a computer installation defines its CICS service level objectives, what they are doing, in fact, is defining response time and throughput goals. These goals can only be achieved and maintained through constant system scrutiny and performance management and tuning.

This discussion will illustrate a logical top down tuning approach to achieve and maintain your CICS service level objectives. First of all, let's define what a top down performance tuning approach is. As we know, CICS executes within the operating system environment. The major objective of the top down tuning approach is to make sure that the operating system environment is well tuned to accommodate CICS before a tuning effort is embarked upon to tune CICS itself. One can vigorously tune CICS internally but if that operating system is not set up to best accommodate CICS, it won't run very well regardless of what you do internally to CICS because that operating system environment will serve as a key limiting factor to CICS performance. Therefore, the operating system must first be in tune, and then whatever changes are made within CICS will make a difference in performance.

Delving further into the top down tuning approach, we can categorize the top down performance tuning procedure into three general areas. Referring to Figure 2 we see operating system factors as the first step in our top down performance tuning effort. The operating system factors fall into three general categories. First and most importantly, DASD or I/O) contention; secondly, paging contention, that is, CICS contending with other jobs on the system for real storage; and thirdly, CPU contention and availability. We will address each of these areas in detail later on as we go through our performance tuning effort.

The second general area of our top down performance tuning approach are address space factors that affect CICS performance. Our scenario will profile a CICS system running under the MVS operating system. In address space area we will examine the various components of the C address space and optimize these components in order to provide ma CICS availability and performance. We will look at the Local Syst Queue Area, or LSQA, we'll examine the OSCOR area, the CICS nucleu and the Dynamic Storage Area, or DSA.

The objective of tuning the address space factors is to optimize the areas that comprise the CICS address space in order to assure maximum availability, reliability, response time, and throughput. Again, all these factors add up to achieving your service level goals. After the address space components are addressed, we will examine the third and last general area in top down performance tuning approach, and those are CICS internal factors. Firstly, we'll look at limits which are primarily CICS transaction control factors, and we'll see how these factors play a part in optimizing your CICS performance. We will then examine some timings which will affect not only CICS's communications with the operating system as well. Finally, we'll examine CICS priorities and we'll be viewing priorities in two ways. First of all, where CICS fits into the operating system, and secondly, transaction priorities within CICS itself. Let us now look at the top down performance tuning areas in depth.

OPERATING SYSTEMS FACTORS

I/O Contention

On most CICS system, I/O contention is the major CICS performance bottleneck. On the average, 75% of CICS performance problems are cause by I/O and DASD not being properly optimized. When we look at I/O contention, we have to examine two factors: the number of I/Os and the contention, we have to examine two factors: the number of I/Os and the duration of these I/Os. Let's first address how to reduce the number of physical I/Os within CICS. This is usually the greatest factor affecting response time and throughput. Since the vast majority of CICS data files are VSAM KSDS, you will generally

find two or three high level indexes, known as the control area, or CA. If a VSAM data record is requested by CICS to a file with three high level indices, and if none of those index records are already within a VSAM buffer in storage, then it would take five physical I/Os to retrieve that data record. That is, one I/O to each of the three high level indices, one I/O for the sequence set, and finally one I/O to retrieve the data record.

Let's assume that this VSAM file exists on a 3380 storage device. We could assume an average service time for each I/O of about 30 milliseconds. If each of your CICS transactions is doing multiple I/Os, this could seriously impact response time and throughput, and certainly cause you to fall short of achieving your service level objectives.

A CICS facility that should be used to address this issue is VSAM files that are to be included in LSR. The end result is having multiple VSAM files sharing relatively large buffer pools. One of the functions of VSAM is to do look aside processing. When a VSAM record is requested, before it does a physical I/O to the requested file, the buffer pools are searched to see if that record already resides in storage. If the record is found in storage, thus saving a physical I/O, this is called a successful look aside. This is especially important when retrieving index records because index record retrieval makes up the majority of VSAM I/Os; therefore, it is very important to define a sufficient number of index buffers within the LSR pool, so that the look aside hit rate would run around 90%. This will substantially cut down on the number of physical I/O's and greatly increase response time and throughput. However, there's no free lunch.

The trade off of saving physical I/Os is increasing CPU cycles because that particular buffer pool that corresponds with the record size that VSAM is trying to retrieve will be referenced by VSAM before the physical I/O is done. The net result is an increase in CICS overhead due to buffer referencing requirements. However, the trade off is usually very positive. (Figure 3 shows CPU time in seconds for 1,000 CICS transaction, each doing five reads. Figure 4 shows the corresponding decrease in response time.) As you can see, response time decreased by .9 seconds on the average to .4 seconds, a net savings in response time of more than 50%. And as you can see by these figures there's also a point of no return. On this particular system, there was not much improvement after specifying 100 LSR buffers in the index buffer pool. This is one of the major methods to reduce physical I/Os and, therefore, greatly improved response time and throughput.

Reducing Paging

Now let's look at one highly recommended method for reduction of CICS paging with an MVS. This method is known as storage isolation. The MVS system resource manager, or SRM, controls access to system resources. These resources include real storage. The control id done by swapping address spaces in or out of storage based on workload and allocation of resources. It's highly recommended that CICS be designated as non-swappable. Input to the SRM is specified in the installation performance specifications, or IPS member of SYS1.PARMLIB. The performance analyst divides the types of work in the IPS into distinct groups, assigns relative importance to each group, and provides performance characteristics for each address space within each group. the OPT member of SYS1.PARMLIB contains parameters that affect paging decisions by the SRM. Storage isolation is specified right here through the PPGRT parameter, PWSS. and the MVS/SA, PPGRTR. Therefore, through specifications within the IPS and OPT, CICS paging can be kept down to a minimum.

CPU Availability and Utilization

One of the common problems that cause CICS to have lack of CPU availability is the priority of CICS within the operating system scheme of things. It is recommended that CICS be placed high in priority within the operating system, right below VTAM but above JES. On a multi-region operation, or MRO CICS system, the terminal owning region should be placed higher in priority in the application owning region, which in turn should be placed higher in priority than the resource, or file owning region. This will cause the CICS terminal network to be serviced quicker, thus improving response time and throughput. If we examine an MVS address space with an MVS370 and MVS/XA below the 16 megabyte line, we'll find that approximately 8 megs or half that address space is being utilized by MVS common, or shared areas. These areas include the MVS nucleus, the MVS common systems are (or CSA), link pack area (or LPA), and System Queue Area (or SQA). The remaining portion of the address space in this example, 8 megabytes, is called the private area which is used by the individual job running within that address space. Looking at Figure 5, we can see the layout of an MVS address space and a typical CICS private area. We see in this particular instance that we have a CICS system running DL1, the bottom block shows an IMS nucleus.

The major components within this address space consists of LSQA, or Local System Queue Area, which is the difference between the high address of the private area, and the top of the region that is specified int he CICS job step. Technically, LSQA consists of MVS

subpools 253-255. However, there are two other areas that fall within this space, the System Work Area (or SWA) which are MVS subpools 236 and 237,and two other subpools, 229 and 230. One of the major users of subpool 229 and 230 is VTAM. VTAM uses this area for its buffering in CICS. The MAXPVT parameter can limit the amount of this buffering. A common symptom of running out of LSQA is an S40D abend. The way to remedy this situation is to effectively increase the size of the private area by either reducing MVS common area usage, such as removing modules from the link pack area, or decreasing the size of the CICS region. Looking at the CICS region itself, we can see that it is comprised of the CICS nucleus, which contains CICS management module s and tables, a resident application program area, OSCOR, and the dynamic storage area.

OSCOR is an areas set aside within the CICS region for MVS get-names after completion of CICS initialization. Running out of OSCOR will cause CICS to abend with an S90A. Be sure that you have enough OSCAR to accommodate your needs, but not too much OSCOR so that space is just being wasted within the CICS region. The objective here is to optimize both the size of LSQA and OSCOR to provide a sufficient amount of DSA so that your CICS system will almost never go short on storage and will also incur very few programs compressions. Let's now look at some of the CICS internal factors that will affect response time and throughput.

CICS Internal Factors

Let's examine two limiting factors within CICS: Active Maximum Tasks (or AMXT), and Max Tasks (or MXT). AMXT is the key system limiting factor within CICS. AMXT tells the CICS task control facility how far down the active task chain to interrogate before terminating its scan. This has a profound effect on a storage requirements and working set size. MXT is a limit for CICS as to how many total transactions may be running within CICS. It is AMXT, however, that is the key system limiting factor and AMXT should be used to control the throughput and storage requirements within CICS. After a total top down performance tuning procedure is followed, AMXT is the last parameter that should be set for the best response time and maximum throughput. Setting AMXT lower than it should be would be putting a false limit on overall CICS performance. As we raise the MXT limit, our throughput along with our working set size will increase, and then we'll reach a point where as we continue raising AMXT other system facts will start to degrade CICS performance, such as going short on storage. The AMXT setting should then be backed up to the point which will provide the best response time and throughput. One of the major timing factors that affect CICS performance is the region exit time interval (or ICV value). CICS waits on an ECB list to the operating system. One of these ECB's is a 'timer ECB, which is the ICV value. The other ECB's on CICS is waitlist to the operating system include file ECB's. If this ICV value, for example, is set in the system initialization table at one second, when CICS issues its wait to the operating system, this ICV timer starts to run. If no other event completes within this one second interval the ICV timer will expire, the ECB will be posted and CICS will request control back from the operating system. Specifying this ICV value at one second or less will assure that CICS gets control frequently enough from the operating system to have a positive impact upon response time and throughput. This is especially important on a lightly loaded CICS system where file I/Os may tend not to complete within a one second time frame. This will assure that CICS will have most of its working set resident and not be severely affected by page in operations when it receives control back from the operating system.

Finally, let's look in the example of priorities within CICS. There are three factors that make up dispatching priority of a CICS transaction., These three factors are transaction priority from the program control table (or PCT), operator priority from the signon table (or SNT), and terminal priority from the terminal control table (or TCT). Some of these factors add up to total dispatching priority and cannot exceed a priority of 255. It's important that high volume inquiry transactions that are in and out of the CICS system very quickly be given a higher priority than update transaction, which might take longer to run and have a lower transaction volume than inquiry transactions.

There are, of course, many more factors and parameters that impact CICS performance. The primary purpose of this article was to show a top down performance tuning methodology that should be followed in assuring optimum CICS performance.

SECTION 3 - PLANNING FOR FUTURE GROWTH

Looking forward towards the 1990's, we'll see an acceleration of the continuing evolution of system architectures. CICS, as IBM's strategic transaction processing system, will play an increasingly major role in this evolution. Let's examine some of the Information Processing industry trends, and how CICS strategies fit into the big picture.

As we more towards the '90's, 'direct service' applications such as home banking and point of sale funds transfers will comprise an increasing share of the transaction processing workload. (Figure 6

illustrates the trend toward direct service applications.) as you can see by 1990 approximately 70% of the total transaction processing workload will be comprised of 'direct service' applications. To facilitate this incredible growth both hardware and software, specifically CPU's and operating systems will evolve to handle this increase in terminal network growth. For example, in 1980, CICS was capable of supporting a 5,000 terminal network with a maximum throughput of approximately 30 transactions per second. IBM projects that in 1990 networks will be over 100,000 terminals with a transaction rate of greater than 500 transactions per second. Operating systems and processors will evolve to handle this workload. CICS systems, of course, must evolve as well. Some of the industry trends affecting the direction the direction that CICS will take are application evolution, along with decentralized operation and connectivity between like and unlike systems.

In the application area, we'll see a tremendous growth of revenue earning rather than just cost savings, 'direct service' applications. Because of these 'direct service applications, the typical end user will be a non-computer oriented consumer. Fourth generation languages (4GL) will help to provide the user friendly interface for these users. As the cost per MIP decreases, application development costs consume a higher percentage of the DP budget. We'll see an increasing use of packaged applications and application generators to help reduce these costs.

Decentralization operations which include local area networks, intelligent workstations, and departmental systems such as the 9370, are other factors in accelerating the growth of CICS. Working hand in hand with decentralized operations, is the need for connectivity between similar and unlike systems, we will see Open Systems Interconnect (OSI) become a standard in the industry. As CICS evolves to support these factors, protection of CICS customers existing applications must be considered. CICS will remain upward compatible from release to release, along with migration tools to ease the pain of change. As MIS gains more direct exposure to the consumer marketplace to 'direct service' applications, CICS performance, availability, and continuous operation factors become increasingly important.

CICS is continually evolving to provide high system availability as well as getting closer to the goal of continuous operation. CICS Release 2.1 wit the extended recovery facility (XRF) which allows a switch to a hot standby CICS system should the primary system run into availability problems. This is an excellent example of IBM's commitment to work toward a high availability and continuously operating CICS system. CICS will, of course, keep evolving to support new operating systems as well as CPU's, so that these high transaction volumes can be realized.

IBM's overall objective is to ensure that CICS is the major transaction processor for the 1990's. Because CICS plays such a major role in IBM's business projections, IBM is working hard to ensure that CICS provides the performance availability, continuous operation, capacity, and the functions to support their CICS customers future business requirements. And because of the heavy emphasis that IBM is placing upon CICS strategy, companies with heavy transaction process requirements should consider CICS as the only way to go.

In summary, we've seen some phenomenal growth in CICS systems over the last 20 years. However, these past 20 years growth will be dwarfed by the revolutionary advancements and achievements we can expect of CICS in the next ten years. The emphasis of CICS development will be on high availability and continuous operation, support of very large networks with high throughput interconnectivity for on like systems through OSI architecture, improved data recoverability and integrity, application development improvement, 4GL support, and CICS with DB2 as a viable production DB/DC system.

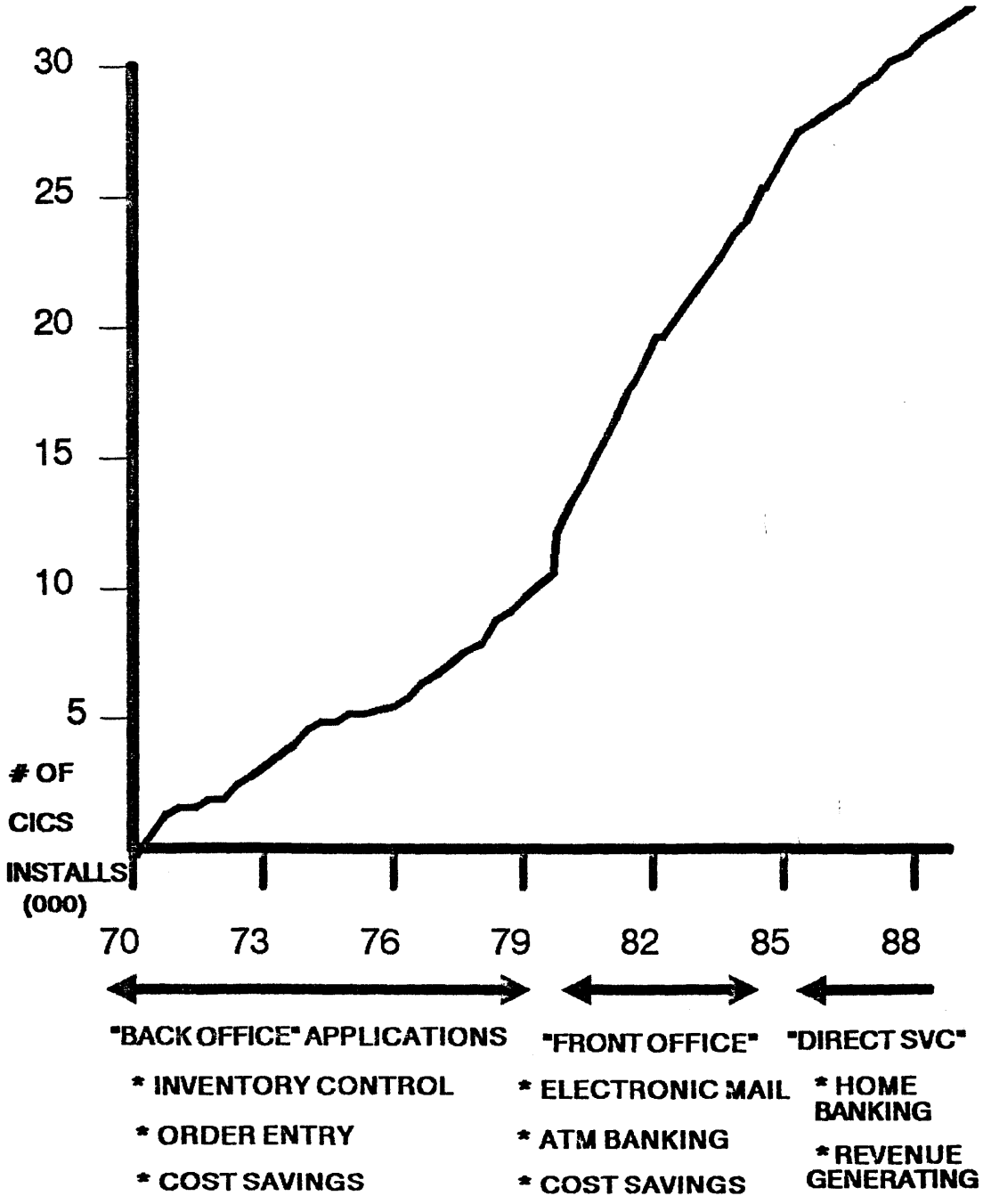


FIGURE 1: CICS INSTALLATIONS (WORLDWIDE)

TOP DOWN PERFORMANCE TUNING

- OPERATING SYSTEM FACTORS
  - I/O CONTENTION
  - PAGING
  - CPU
- ADDRESS SPACE FACTORS
  - LSQA
  - OSCOR
  - DSA
- CICS INTERNAL FACTORS
  - LIMITS
  - TIMINGS
  - PRIORITIES

FIGURE 2

CMG3 09/04/87

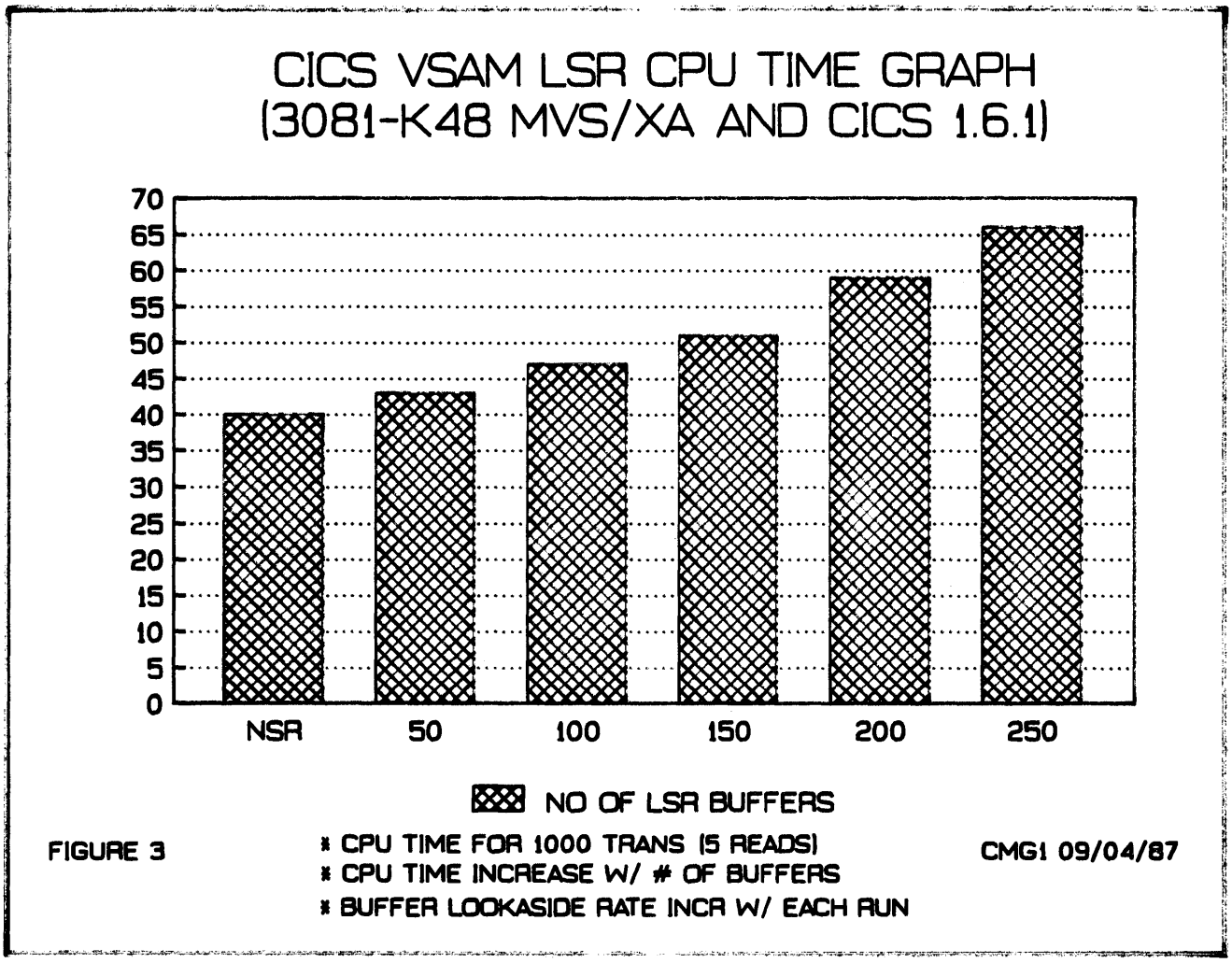

CICS VSAM LSR CPU TIME GRAPH
(3081-K48 MVS/XA AND CICS 1.6.1)
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
NSR
50
100
150
200
250
NO OF LSR BUFFERS
* CPU TIME FOR 1000 TRANS (5 READS)
* CPU TIME INCREASE W/ # OF BUFFERS
* BUFFER LOOKASIDE RATE INCR W/ EACH RUN
CMG1 09/04/87

FIGURE 3

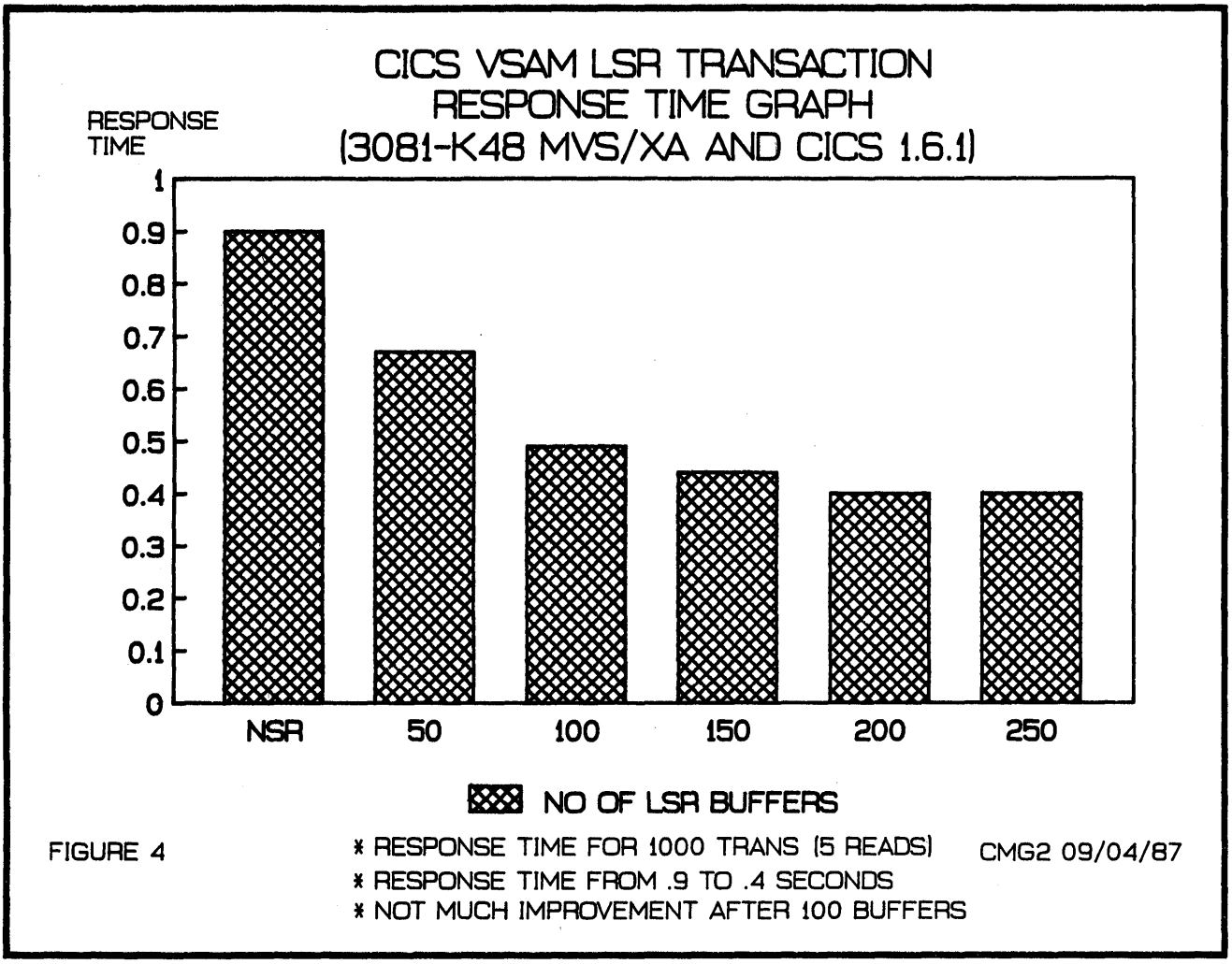


FIGURE 4

* RESPONSE TIME FOR 1000 TRANS (5 READS)
* RESPONSE TIME FROM .9 TO .4 SECONDS
* NOT MUCH IMPROVEMENT AFTER 100 BUFFERS

CMG2 09/04/87

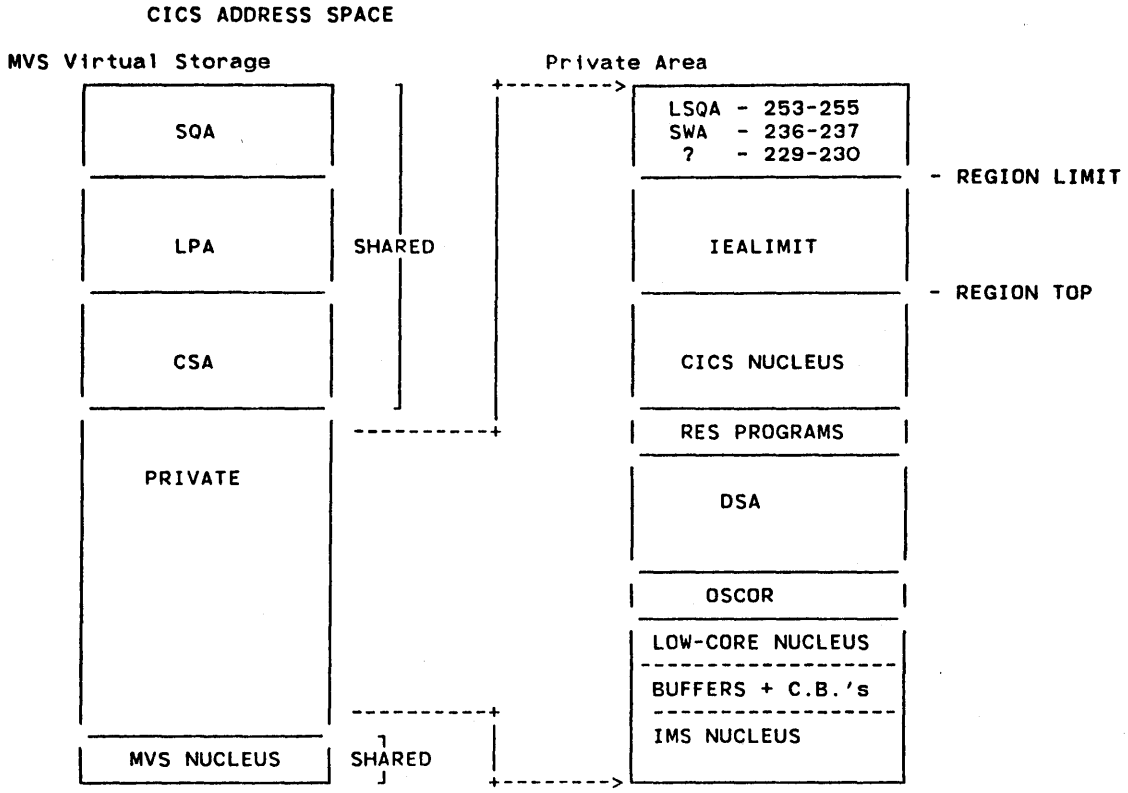

CICS ADDRESS SPACE
MVS Virtual Storage
Private Area
SQA
LPA
CSA
PRIVATE
MVS NUCLEUS
SHARED
SHARED
LSQA - 253-255
SWA - 236-237
? - 229-230
- REGION LIMIT
IEALIMIT
- REGION TOP
CICS NUCLEUS
RES PROGRAMS
DSA
OSCOR
LOW-CORE NUCLEUS
BUFFERS + C.B.'s
IMS NUCLEUS

Figure 5

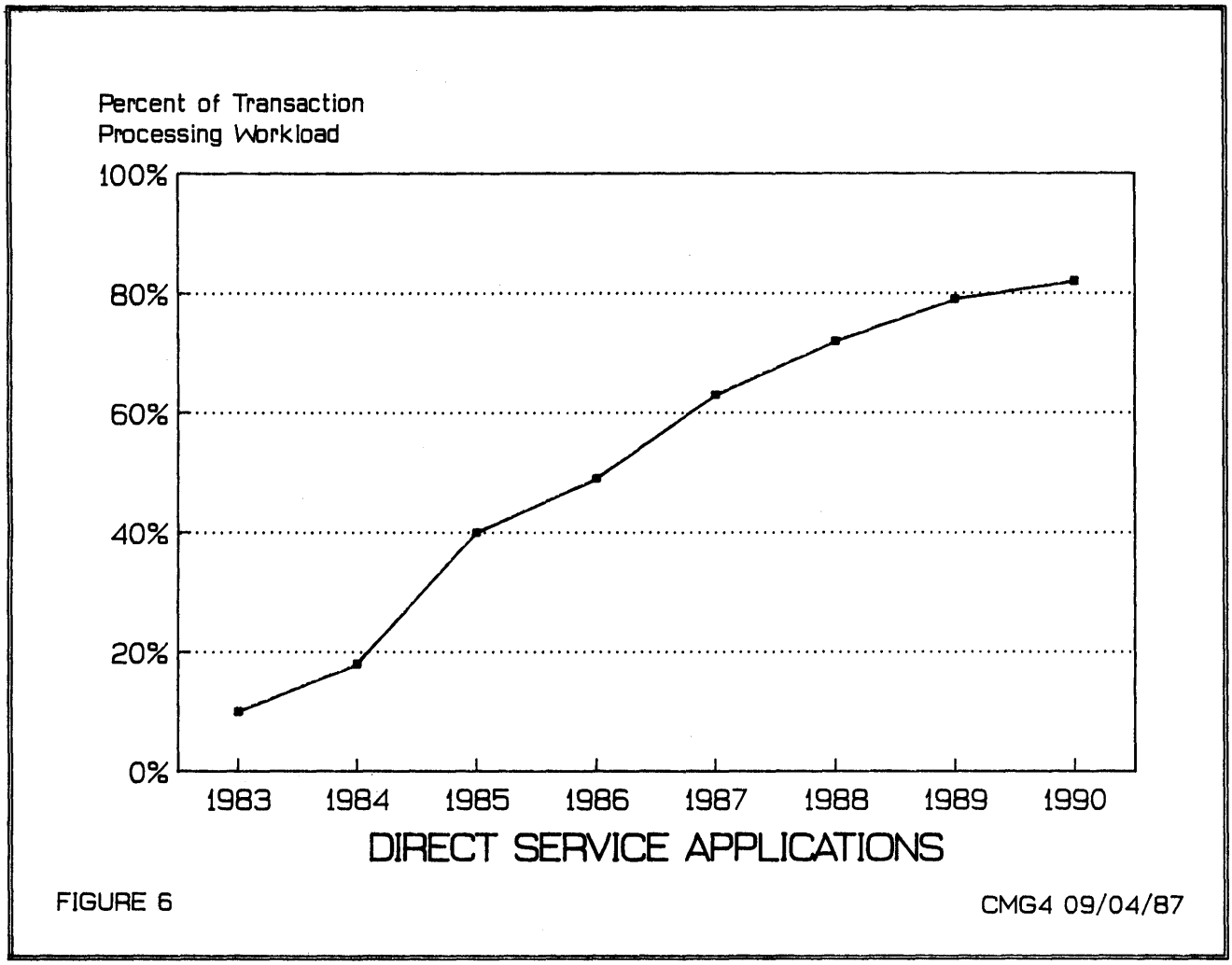


FIGURE 6

CMG4 09/04/87

RULES OF THUMB

Computer Measurement Group (CMG) Ontario Inc.

Harry Zimmer
Amdahl Canada Limited

ABSTRACT

Performance and Capacity Planning Managers have three methods available to them on how to do and validate plans. (1) "The Benchmark", which is typically extremely expensive and time consuming. (2) "Back of the Envelope", guessing done in under ten minutes. (3) "Rules of Thumb", a middleground between benchmarking and guessing. Our experience in the Regional CMG Group in Ontario, is that given todays pressures of being fast, cheap and accurate, RULES of THUMB are the Planning Managers' best alternative.

Table of Contents

1. Introduction

2. General Rules of Thumb
   - 2.1 Introduction
   - 2.2 Performance
   - 2.3 Capacity
   - 2.4 Forecasting Workloads
   - 2.5 General Rules of Thumb
   - 2.6 Assumptions

3. MVS
   - 3.1 MVS General Rules of Thumb
   - 3.2 MVS Performance
   - 3.3 MVS Capacity
   - 3.4 MVS Data Bases
     - 3.4.1 DB2
     - 3.4.2 IMS
     - 3.4.3 IDMS

4. VM
   - 4.1 VM General Rules of Thumb
   - 4.2 VM Performance
   - 4.3 VM Capacity

5. Tandem
   - 5.1 Tandem General Rules of Thumb
   - 5.2 Tandem Performance
   - 5.3 Tandem Capacity

6. Data Communications
   - 6.1 Response Time Analysis
   - 6.2 Transmission Volumes
   - 6.3 General Data Communications Rules of Thumb

7. Printing
   - 7.1 Printing General Rules of Thumb
   - 7.2 Impact Printing Technologies
   - 7.3 Non-Impact Printing Technologies

1. Introduction

The purpose of this document is to:

1. Establish general and specific capacity guidelines.

2. Establish general and specific performance guidelines.

3. Establish several methods for estimating future processing workloads.

Contributions to this document were made by several CMG-Ontario members. A list of contributors appears on the next page. I would like to extend a special note of thanks to Mike Hurley for the help that he provided in the production of this document.

This document is an updated version of one published originally in September 1985, and updated in September 1986 and April 1987 (CMG Spring Transactions).

The material presented in this document should be used for planning purposes only. The rules are general in nature, and specific installations may experience large deviations in specific cases.

Submissions and updates for this document are welcomed - and encouraged. If you would like to make a submission, or if you have any questions or comments about this document, please contact me in writing:

Harry Zimmer
Regional Systems Engineer
Amdahl Canada Limited,
1 First Canadian Place, Fifth Floor,
Toronto, Ontario, Canada M5X 1A4

List of Contributors:

Leilani Allen	Amdahl Corporation
Bob Ashton	DB-VIEW Inc.
Ernie Crawford	Xerox Canada Inc.
Gabe Gewurtz	Ontario Hydro
Mike Hurley	Amdahl Canada Limited
Keith Leung	Tandem Computers Canada Limited
Doug Lewars	Canada Packers
Lorraine Martel	LGS Data Processing Consultants Inc.
Drew McKenzie	LGS Data Processing Consultants Inc.
Anthony Mungal	Amdahl Canada Limited
Geoff Senson	Amdahl Canada Limited
Ken Strauss	Ontario Hydro
Harvey Warner	BGS Systems, Inc.

2. General Rules of Thumb

2.1 Introduction

Here are some general points of reference which should be understood before using this document.

Users can be classified into one of the following major two categories:

1. Operational Systems

* TSO Primarily (ISPF)
* CICS/S2K, CICS/IMS, CICS/IDMS
* Batch Users
* Any combination of the above

2. Management or Decision Support Systems

* PROFS, Personal Services/370
* DB2, SQL
* SAS Family, APL, ADRS - 4GL's
* Any combination of the above

Each of these two user groups requires in depth investigation and attention from data processing performance and capacity planning departments. Here is a list of the critical issues:

* Business Objectives
* Availability
* Responsiveness
* Growth
* Cost Effectiveness
* Service Levels
* Stability
* Quality
* Productivity
* User Satisfaction

2.2 Performance

Processing performance is defined by the following components:

* Online Response Time
* Interactive User Response Time
* Batch Throughput
* Batch Deadlines (Turnaround Time)
* Backup and Recovery Window

These components are influenced by the following:

* Processor - CPU Utilization (Average and Peak), CPU Processing Speed.
* Main Storage - Storage Utilization and Paging Rate
* I/O Dasd Subsystem - Paging Rate, I/O Rate by Channel, I/O Rate by Device, and Wait State by Device.
* Data Communications - Protocol Complexity, Line Speed, Traffic Complexity, Network Topology, Geographical Distance, and Network Transmission Technology.
* Software Efficiency - Operating System, Program Products, Custom Designed Applications.

Some general performance rules of thumb are:

1. 80% of the improvement results from 20% of the work.
2. Left to themselves, things always go from bad to worse.
3. If there is the possibility of several things going wrong, the one that will go wrong is the one that will do the most damage.
4. All CPU's wait at the same speed.

2.3 Capacity

Workload processing capacity can be described by the following parameters:

* Processor (CPU) Speed - Number of Cycles Available, Number of Concurrent Engines at Work, and queue length.
* Main Storage Size - Number of Megabytes of Space Required vs Page Swap Rate. Main Storage, Expanded Storage and Secondary Storage relationships.
* I/O - DASD Subsystem Size and Throughput - Number of volumes of DASD, Number of Tape Drives, Number of Channels, and Number of Paths.
* Data Communications - Number and type of front end communications ports, Front End Processor Speed, Line Speeds, and remote device controller speed and memory size.

2.4 Forecasting Workloads

When forecasting workloads, the following factors should be considered:

* Growth/Reduction in Natural Business Units (eg. Growth in number of credit card transactions per month).
* System Enhancements (eg. Add three new transaction types.)
* Expansion (eg. Increase in the number of new users and number of locations. Of particular concern are mergers and acquisitions.)
* Change in hours of operation.
* Change in reporting frequency (eg. Change from monthly to weekly reporting, or with inquiry packages, from standard to ad hoc.)

For systems in production, future resource requirements should be extrapolated based on:

* Actual Resource Utilization (eg. Number of CPU hours per month.)
* Natural Business Units (eg. Number of Credit Card transactions per month.)

It is sometimes useful to do a forecast based on historical data. The most commonly used techniques are (please consult a statistics manual for the formulae):

* Simple Moving Averages
* Exponential Smoothing
* Classic Decomposition
* Simple Linear Regression
* Multiple Linear Regression
* The Box-Jenkins Method

There are 4 general trends which may be observed:

1. TREND data pattern - when the workload increases as a result of changes in the way the existing system is used. Eg. An increase in the number of transactions processed by a system.
2. CYCLICAL data pattern - which can be caused by a system change, such as the introduction of new applications and the elimination of others.
3. SEASONAL data pattern - which may be caused by periodic tasks that effect the overall system, such as year end or month end processing.
4. STATIONARY data pattern - when a time series behaves as random fluctuations around a mean value, and no trend can be identified.

2.5 General Rules of Thumb

1. The maximum number of active users on a system at any one time is equal to the number of user-id's divided by two.
2. Assume a peak to average ratio of 2.25 : 1.
   This is based on (a) Peak Hour to Average Hour Ratio being 1.5 : 1, and (b) Peak Day to Average Day Ratio being 1.5 : 1.
3. Peak utilization should never exceed 90% (therefore using a peak to average ratio of 2.25 : 1, the average utilization should never exceed 40%).
4. Personal Computers are assumed to behave as terminals, but will consume 25% more than terminals in CPU utilization.
5. Dial-Up users will, on average, consume 25% less in CPU utilization.
6. Prime Shift is defined as 07:00 to 19:00, excluding weekends and holidays. Work done in hours other than this period is classified as Non-Prime Shift Work. This definition is most applicable for company workloads which service Canada on a coast to coast basis.

7. Workload distribution of a system can be analyzed based on the number of programs a company runs:

# of Programs	% CPU Usage
2% of the total uses	50% of the resource
48% of the total uses	48% of the resource
50% of the total uses	2% of the resource

8. A commonly used measure of CPU usage and capacity is "the number of CPU hours consumed" versus the "number of CPU hours available" in a processing complex (CPU). A simple and fast method for calculating the capacity of a CPU is described by the following formula. This calculation is independent of the vendor, CPU model, and operating system:

Capacity # of CPU HRS = # of business days/month * # of hrs/day * % utiliz for good service * # of internal engines

Definitions:

# of business days/month	=	Depends on the month and the number of holidays. It generally averages out to 20 business days for an average Ontario company.
# of hours/day	=	Pick a measurement window - this is called the "Prime Shift". This is typically 7am to 7pm (12 hours).
% utilization for good service	=	80% is typically used by CMG-Ontario member installations.
# of internal engines	=	Today's CPU complex's typically have 1, 2, 3, 4 or 6 internal processors. (eg. 3090-200E 2 engines, and the 5890-190E has 1 engine.)

Sample Calculation #1

IBM 3081KX - 8 hrs/day - 20 business days - 80% utilization maximum. The 3081KX has 2 internal engines.

Max 256 Hours of 3081KX CPU = 20 days/month * 8 hrs/day * 0.8 * 2

Sample Calculation #2

Amdahl 5860 - 20 hrs/day - 18 business days - 90% utilization maximum. The 5860 has 1 internal engine.

Max 324 Hours of 5860 CPU = 18 days/month * 20 hrs/day * 0.9 * 1

2.6 Assumptions

1. The CPU is assumed to be the key resource.
2. All CPU estimates are expressed in TRUE CPU time (SMF with a capture ratio applied).
3. The system will be configured with no bottlenecks.
4. Online workload is the critical service to plan for.
5. The Prime Shift is the critical window to plan for in all operating environments. (ie. Use you company's peak processing window and plan around it.)
6. Network Response is defined as: Front End Communications Processor to Terminal Response. An average cross-network response time should always be planned for with the understanding that actual response times for locally attached and inter-city traffic may be faster.
7. The sources of the Rules of Thumb in this document are widespread and include:
   * Production Experience
   * Benchmarks
   * Simulations
   * Analytical Queuing Models

3. MVS

3.1 MVS General Rules of Thumb

For planning purposes, MVS users may be classified into one of four categories:

1. TSO Primarily (ONLINE)
2. CICS/IMS, CICS/IDMS, CICS/S2K, or CICS/DB2 Primarily (ONLINE)
3. Batch Primarily (BATCH)
4. Combination of Services (ONLINE+BATCH)

The basic data resources for doing MVS performance and capacity planning are SMF, RMF, and PA II data. In general a facility will produce 1,000,000 records of SMF data per CPU mips per month.

3.2 MVS Performance

Here is a list of the MVS performance monitor points, and some performance objectives:

Component	Objective
CPU	
CPU Busy	GT 85%
A.S. IN&READY ....see N1	N/A
A.S. OUT&READY	0
A.S. LOGICAL OUT&READY	0
Batch A.S. ....see N1	N/A
TSO A.S. ....see N1	N/A
STORAGE	
Minimum Available Frame Queue	GT 15
Average Available Frame Queue (Real Storage)	GE 4% of real storage
UIC - Over Committed	LE 2
- Under Utilized	GT 64
PAGING	
Page Fault Rate ....see N2	LT 25 Pages/Sec
Total Paging Rate ....see N3	LT 120 Pages/Sec
3330/3350 Page Rate	12-15 Pages/Sec
6280/3380 Page Rate	16-18 Pages/Sec
2305 Page Rate	25-30 Pages/Sec
S.S.D. Page Rate	100-250 Pages/Sec
PLPA/CSA Page Rate	LT 5 Pages/Sec
Paging Device Activity	LT 30% Busy
Page Service Time	LE 25 msec
SWAPPING	
Input Terminal Wait	GT 85% of All Swaps
Output Terminal Wait	LT 5% of All Swaps
Detected Wait ....see N1	N/A
ENQ Exchange	0
Unilateral	LT 5% swaps
Exchange	LT 5% swaps
TSO Swap-In Size (with swap data sets)	36-72 Pages
TSO Swap-In Size (with local d.s. only)	60 or 90 Pages
Swap/Transaction Ratio (TSO)	LT 1.1 to 1.0
Logical Swap Out	GE85% of Eligible Swaps
Logical Swap In	GE95% of Logical Swapouts
Swap Set Service Time (3350)	60-80 msec
(3380)	35-50 msec
CHANNELS	
Selector Channels (Tape)	LT 60% Busy
Block Multiplexor Chans - DASD	
370 Mode or NO DPR ....Batch	LT 30-40% Busy
TSO	LT 30-35% Busy
Online	LT 25-30% Busy
XA Mode - 1 path & DPR ....Batch	LT 30% Busy
TSO	LT 30% Busy
Online	LT 25% Busy
XA Mode - 2 paths & DPR ....Batch	LT 54% Busy
TSO	LT 54% Busy
Online	LT 50% Busy
XA Mode - 4 paths & DPR ....Batch	LT 74% Busy
TSO	LT 74% Busy
Online	LT 70% Busy
RPS Factor ....see N5	LE 3.5 & GE 2.5
Channel Busy and CPU Wait	LE 10% of Channel Busy
Channel Average Queue Length	LE 0.1

DEVICE

Device Busy	LT 30%
Time Per Access 3380 ....see N6	LT 30 msec
Time Per Access 3350 ....see N6	LT 40 msec
Avg Queue Length / Device Busy	LT 0.05
Avg Service Time (370) 3380 ...see N7	LT 35 msec
Avg Service Time (370) 3350 ...see N7	LT 45 msec
Average Response Time (XA) 3380	LT 35 msec
Average Response Time (XA) 3350	LT 45 msec
IO S.Q. Time (XA)	LT 5 msec
Seek Distance	90% in 10 Cyls
Zero Seeks	GE 50% of total

TSO Guidelines

Swap/Ended Transactions (First Period)	LE 1:1
Avg Time Ended First Period	LE 1 sec
Avg Wait to Start (First Period) ..N8	LT 0.5
% Ended in First Period	90-95%
% Ended in Second Period	5-8%
% Ended in Third Period	2-4%
TSO Trans Rate (based on 50 users) N9	GE 1.5 per sec

MISCELLANEOUS

Number of Domains Defined	LT 10
RMF Reporting Interval ...(in minutes)	GE 15 & LE 30
RMF Sample Cycle	GE 250 msec & LE 1000 msec
DASD Block Size (Random)	up to 1/2 track size
DASD Block Size (Sequential)	up to full track (except 3380)
(a) Mixed Devices (3330/3350/3380)	About 6K (LE 6223)
(b) Mixed Devices (3350/3380)	About 9K (LE 9076)
VSAM Block Size	4K
Tape Block Size	8-12K minimum
6250bpi	up to 32K

MVS Performance Objectives Notes

N1 - Installation Dependent - can be useful for capacity planning
N2 - Sum of Non-VIO Reclaims + Non-swap, Non-VIO Page-Ins.
Installation and CPU Dependent - V/7B or 3083E assumed.
N3 - Sum of all Page-Ins + all Page-Outs.
Installation and CPU Dependent - V/7B or 3083E assumed.
N4 - 370-mode or No DPR assumed.
N5 - <Total Device Busy On Chan> divided by <Channel Busy>
N6 - <Device Busy> divided by <Activity Rate>
N7 - [<Device Busy> plus <Queue Length>] divided by <Activity Rate>
N8 - <Ended Time> - [<Avg Trans SVC Rate> * <Ended Time> divided by <Absorption rate>].
N9 - <Total Transactions> divided by <Report Interval>.

Acronym Definitions:

LE = "Less than or equal to"
LT = "Less than"
GE = "Greater than or equal to"
GT = "Greater than"
N/A = Not applicable

Note: Performance Values cited typically apply to 15 minute intervals. Care should be taken when trying to apply performance objectives over short periods of time.

DASD Controller Guidelines

* CACHED *	Interactive Workloads	Batch Workloads
I/O Rates <1>	< 160	< 180
Hits Ratio <2>	> 90 %	> 65 %
Read/Write Ratio	> 3:1	> 2:1
Read Hits/Stage	> 4	> 3
Read Misses/Stage	< 1	< 1

* UNCACHED *	Interactive Workloads	Batch Workloads
I/O Rates <1>	< 45	< 65

<1> Assumes 4K blocksize. Interpolate for blockings < and > 4K. I/O rates = I/O's per second per controller subsystem.
<2> This is Read Hit Ratio.

CACHE Guidelines

For Dedicated Cached Subsystems (i.e. only cached actuators in the subsystem):

(a) Should maintain hit ratios > 85%.
(b) For hit ratios < 85%, disqualify certain datasets or activities and/or Add more cache. Either move these data sets to another volume or out of the subsystem. Locality of reference analysis will be conclusive. (Note: Do not add more cache as the first step of tuning, since the read/write ratio and locality of reference may be the culprits of poor performance. Also see note (i) below.
(c) When hit ratios are consistantly high, channel utilizations can be allowed to increase significantly (up to 50%).

For Selective Cached Subsystems (i.e. 1 or more cached actuators in the subsystem):

(a) Keeping internal path utilization < 35% is critical, otherwise the uncached actuators service time will elongate unbounded, and will result in erratic response times.
(b) For hit ratios > 85%, the internal path utilization will be <= to the channel utilization. As the hit ratio decreases, the internal path utilization will become > the channel utilization.

Note (i): For erratic hit ratios, do the following on a volume basis:

i.1 * Determine Read:Write Ratio (would like this to be greater than or equal to 2:1).
i.2 * Locality of reference, is it good or bad? (Good means I/O's are clustered within a small range of tracks.)

If i.1 is high AND i.2 is good, then add more cache. If either i.1 or i.2 is unsatisfactory, uncache the data set or actuator.

Solid State Disk Guidelines

For 2 directors configured to two 3 megabyte channels:

a) The maximum paging rate is approximately 950 pages/second.
b) Configure one (maximum) page dataset per device address.
c) Configure one (maximum) swap data set per device address.
d) Configure the subsystem on dedicated channels if possible.
e) Two dedicated SSD swap paths will provide higher concurrency than 4 paths (dedicated or shared) to rotating 3380's. This is important for keeping swap-in delay low, and hence maintaining good response times.
f) Page response time: 3 msec.

Disk Guidelines

*** This section to be added in the next rules of thumb document.

3.3 MVS Capacity

The usage of MVS is directly dependent on the type of user making use of the service. In general, the following usage guidelines on a per user basis may be applied:

MVS Workload Forecasting

Type of User	# of CPU HRS/ Month 3081KX	# CYL DASD 3380	# Reels Tape 6250
TSO Primarily	0.250	15	3
CICS-IMS/IDMS/S2K	0.450	25	5
DB2	0.850	26	5
Batch Primarily	0.350	28	7
All Services	0.500	25	5

The CPU values presented in the above figure are totals per month, which includes all shifts (Prime + Non-Prime). The following guidelines may be applied for splitting these totals into prime and non-prime:

* TSO is split 90% prime, and 10% non-prime.
* CICS/IMS, CICS/IDMS, CICS/S2K is split 90% prime and 10% non-prime.
* DB2 is split 95% prime, and 5% non-prime.
* Batch is split 25% prime and 75% non-prime.
* Combination all Services is split 60% prime and 40% non-prime.

MVS CPU and MAIN MEMORY Rules:

For each MIP of installed processing capability, the system will require 4 to 6 megabytes of main memory. For specific dedicated situations:

(a) For IMS/CICS --- 4 megabytes / installed MIP.
(b) For TSO -------- 5 megabytes / installed MIP.
(c) For DB2 -------- 6 megabytes / installed MIP.

Online Workloads

The following information should prove to be of greatest help to systems development managers who require more precise figures for calculating the processing requirements for new ONLINE systems (ie. TSO, CICS/IMS based).

A transaction is defined to be one pressing of the ENTER key. A transaction may be classified as simple, medium or complex depending on the number of data base accesses done by the transaction. The following table defines the guidelines for CPU load forecasting on a transaction level basis:

MVS System Transactions and CPU Load
(In 3081KX CPU seconds)

Transaction Type	Simple	Medium	Complex
Number of I/O's	<=5	<=10	> 10
CICS-IMS/IDMS/S2K	0.038	0.071	0.143
VSAM/CICS	0.008	0.019	0.036
Non DB/DC	0.004	0.008	0.020

If the transaction mix is unknown, then assume 25% simple, 50% medium, and 25% complex.

Batch Workloads

Each online true CPU hour generates one hour of batch processing.

Systems Development

Where no detailed utilization information is available, development and production processing loads can be estimated based on planned development resources using the following guidelines:

Systems Development Workload Forecasting
Average Utilization Per Programmer Per Month

Type Development Tool Used	# of CPU Hours Per Month 3081KX CPU Hours
Conventional (Cobol, PL/I)	0.50
High Level (FOCUS,DB2,ACT/1)	1.00

* For systems which are developed using conventional development tools: one hour of development CPU per month will result in one hour of production CPU load per month twelve months later.
* For systems which are developed using high level development tools: one hour of development CPU per month will result in one hour of production CPU load per month six months later.
* For Online Systems, the resulting production load will be split 50% online and 50% batch.

To determine the prime and non-prime workload, the guidelines are:

* Development (test) and user direct processing load is split 80/20 between prime and non-prime.
* 80% of the online processing takes place during the prime shift.
* 80% of the batch processing takes pace during the non-prime shift.

3.4 MVS Data Bases

3.4.1 DB2

Buffer Manager: Use only one 4K bufferpool. One 4K bufferpool will be managed more effectively than 3 separate bufferpools.

VSAM Files: Use direct VSAM allocation for production tablespaces rather than storage groups. Direct allocation provides better control for tuning DASD devices.

Dispatching Priority: Set the dispatching priority of DB2 address spaces to the highest user priority, ie. higher than the IMS control region. Less than 10% of the DB2 code is executed under this priority and most of it is critical. Most DB2 work is dispatched under the primary address space, such as TSO or CICS.

Data Collection: Collect SMF record type 100 (statistics) and SMF record type 101 (Accounting) on a continual basis for DB2. The DB2 installation parameters SMFSTAT and SMFACCT should be set to collect these records by default. The global trace should not be active.

Resource Control: Release 2.1 of QMF features a user exit which can be used to restrict the resource consumption by user. If QMF is being used with large tables, the exit should be used to restrict the total CPU to avoid "runaway transactions".

3.4.2 IMS

Performance Targets:

IMS Control Region Page-Ins ..... Less than 2 per second

IMS Control Region Page-Faults ..... Less than 10 per second

IMS Message Processing Regions ..... Optimum usage at 60-75% per message region.

Tuning of IMS Systems should be done in the following order:

1. The DASD Subsystem,
2. Virtual Storage,
3. Paging Sybsystem, and finally,
4. Internal IMS Parameters.

3.4.3 IDMS

* Award IDMS the highest reasonable priority in the system (ie just below critical system components like JES and VTAM).
* Make IDMS non-swappable.
* Utilize the storage isolation functions of the SRM to reduce the number of page faults within the IDMS address space.
* Keep the total paging rate to a reasonable level for your paging configuration.

4. VM

4.1 VM General Rules of Thumb

There are three distinct types of VM users:

1. PROFS Primarily
2. Combination All Services
3. SQL Primarily

4.2 VM Performance

Here are the VM Performance monitor points, and some performance objectives:

VM Performance Objectives

Component	Description	Objective
Processor		
RAI	Resource Availability Index	GT 0.8
TOTCPU	Total System Processing	85-90%
TVRATIO	Total to Virtual Ratio	LT 2.0
ACTIVE	Active Users	LT 10%
Q1SEC	Time Between Q1 Drops - Seconds ..	0.10
Q2SEC	Time Between Q2 Drops - Seconds ..	0.50
PRIVOP	Privileged instructions	300 (for 4381-2)
Main Storage		
STGUTIL	Storage Utilization	LT 70%
STEALSCANS.....	Steal Scans	LT 0.10
PAGERATE	Paging Rate	20 pps (for 4381-2)
FREXTEND	Free Storage Extensions	LT 5 (Objective is 0)
I/O Dasd Subsystem		
IOINT	I/O Interrupts	N/A
CHANn	Channel 'n' Utilization	LT 30%
CTLUNIT	Control Unit Busy	LT 30%
DEVICE	Device Unit Busy	LT 30%

LE 5.0 SIO per sec 3350	:	LT 10% Devices with Queue
LE 5.5 SIO per sec 3375	:	LT 50% Seeks are Non-Zero
LE 6.0 SIO per sec 3370	:	LT 40 CYL Average Non-Zero Seek
LE 9.0 SIO per sec 3380	:	

4.3 VM Capacity

The following general capacity rules of thumb apply in the VM world:

* Prime Shift load does not exceed 75-80% of capacity.

* Total all Shifts does not exceed 85% of capacity.

Usage of VM is directly dependent on the type of user making use of the service. In general, the following usage guidelines may be applied:

VM Workload Forecasting

Type of User	# CPU HRS/ Month 3081KX	# CYL DASD 3380	# TAPE REELS 6250
PROFS Primarily	0.100	7	0.1
Comb. Services	0.200	10	0.1
SQL - Occasional	0.100	30	0.2
SQL - Moderate	0.300	30	0.2
SQL - Heavy	0.600	30	0.2

The CPU values presented in the above figure are totals (prime and non-prime shifts). These totals can be split into 90% prime and 10% non-prime.

SQL users have been broken into three classes (if the characteristics are unknown, assume the user to be of moderate type):

1. Occasional - One SQL/DS access per day or less - with minimal requests for summary reports of large data volumes.

2. Moderate - Regular use of an SQL/DS application - with a low to medium frequency of ad-hoc reporting.

3. Heavy - An application developer or end users with a high frequency of ad-hoc report development.

5. Tandem

5.1 Tandem General Rules of Thumb

* To take advantage of Tandem's hardware architecture, applications should be designed so that:

  1. Any application process can be moved to any CPU.

  2. Copies can be made of any application process, and workloads are spread across CPU's.

  3. Any data base file or program file can be moved to any disk drive.

  4. All transaction message-path-lengths are kept to a minimum.

* Based on measurements obtained by the XRAY/MEASURE software monitor, balance workloads across devices of the same type.

5.2 Tandem Performance

Parameter	Recommended MAX Value
CPU.CPU-BUSY-TIME	60% (TNSII), 70% (TXP) (not counting batch)
CPU.CPU-QTIME	1.0
CPU.DISPATCHES	180 per second (TNSII) 300 per second (TXP)
CPU.SWAPS	1 per second
DISC.REQUESTS-BLOCKED	5%
DISC.SEEK-BUSY-TIME	15 milliseconds per request
DISC.READS + WRITES + SEEKS	15 per second (TNSII) 25 per second (TXP)
DISC.READ-BUSY-TIME + WRITE-BUSY-TIME + SEEK-BUSY-TIME	25% (TNSII), 35% (TXP)
DISC.REQUEST-QTIME + PROCESS.REVC-QTIME (disc process)	3
PROCESS.VSEMS (disc process)	1 (DP1 only)
PROCESS.PAGE-FAULTS	2.0 per second

OLTP Performance:

* Standard ET1 (debit-credit banking) transaction
* Hardware: VLX CPU and B30 Operating System
* Pathway with TMF
* 3 random reads with updates (Account, Teller, Branch)
* 1 Sequential write (history)
* Data Base: 200 Million Bytes in size
* Throughput: 10 trans/sec per CPU
* Response Times with 95% less than 2 seconds

5.3 Tandem Capacity

* This section to be added in the next Rules of Thumb Document.

6. Data Communications

6.1 Response Time Analysis

The following guidelines may be used to estimate online response times for most operating environments:

* Response Time = Network Response + CPU Response + DASD Response

* Network Response can be fixed at 2.35 seconds. Before using this consider: line speed, topology & protocol complexity.

* CPU Response is equal to 10 times CPU Usage (eg. if CPU usage is 0.055 seconds, then CPU Response is 0.550 seconds).

* DASD Response is dependent on the number of I/O's. Each I/O takes 0.060 seconds on an IBM 3350, or 0.030 on an IBM 3380. (Eg. For a transaction that has 5 I/O's on an IBM 3380, then the DASD response time is 0.150 seconds).

6.2 Transmission Volumes

The following volumes can be used for planning purposes:

Data ..... 700 char/page
Text ..... 4,000 char/page
Image ..... 35-50,000 char/page
Graphics ..... 50,000 char/page
Voice Annotation ..... 32K - 64K bps

The following Terminal Device load rules of thumb are useful for planning purposes:

Data Communications - Terminal Load Guidelines

Application	Typical Complexity	Trans/Term/Person/Day
Data Entry	Simple	1,000
Inquiry	Medium	300
Update/Inquiry	Complex	500
Personal Computer	Complex	100
Word Processing	Complex	100

General Data Communications Rules of Thumb:

(1) For an SNA/SDLC environment:
A 9600BPS line can support 3 or 4 3270 type controllers. Each controller is assumed to be able to support up to 32 devices. Therefore a 9600BPS line can support from 96 to 128 devices. Use 96 as a maximum for high production traffic, and 128 as a maximum for lower development/test traffic.

(2) Error Rates: 1 error in every 10 to the power 5 bits is considered to be acceptable. Greater than this is considered to be unacceptable.

7. Printing

7.1 General Printing Rules of Thumb

Most printers have unique operating and capacity characteristics. In general, the following attributes may be applied to all users of printers:

(a) Average # of Lines Per Page ==========> 60

(b) Average # of Characters Per Line ==========> 120

(c) Average # of Characters Per Page ==========> 7,200

SPEED Comparisons

Lines/Minute (LPM)	Pages/Minute (PPM)
600	10
1200	20
2400	40
3600	60

7.2 Impact Printing Technologies

* Average forms mount time = 10 minutes.

7.3 Non-Impact Printing Technologies

* %'age of printed output that can be printed 2-up AND duplexed is 30% of total data centre print volume.

This document provides a vehicle for sharing techniques and experiences in the fields of computer systems measurement, evaluation, capacity management and planning.

NO GUARANTEE or liability of the ideas and concepts set forth in this publication is assumed by the authors, their employers, or the Computer Measurement Group (CMG) Ontario Inc. <CMG-Ontario>, it's members and executive, and their employers.

CMG-Ontario is a regional group of the Computer Measurement Group Inc. (CMG), which is an international non-profit corporation based in the United States.

MEMBERSHIP - If you are not currently a member of CMG-Ontario, please drop our membership chairman a note, and he will be pleased to send you a membership application form. Membership is currently $45 per year per person, or a corporate membership is available for $180 per year. Please contact:

Foy Chong
Senior Technical Analyst,
Toronto Dominion Bank,
Central Computing Facility,
P.O. Box 1, Toronto-Dominion Centre
Toronto, Ontario, Canada L5M 2H3

CONTRIBUTIONS - to this rules of thumb document are welcomed and encouraged. All submissions are subject to editing. Please mail all submissions to the Rules of Thumb Editor:

Harry Zimmer
Regional Systems Engineer
Amdahl Canada Limited,
1 First Canadian Place, Fifth Floor,
Toronto, Ontario, Canada M5X 1A4

Permission is hereby granted to members of CMG - Ontario to reproduce this publication in whole or in part solely for internal distribution within the members organization, provided that the copyright notice is set forth in full text on the title page of each item reproduced.

RULES OF THUMB is published by the:
Computer Measurement Group (CMG) Ontario Inc.

COPYRIGHT C 1987 by:
Computer Measurement Group (CMG) Ontario Inc.
Toronto, Ontario, Canada.

GOOD TOOLS AND MODELS ARE NOT ENOUGH: ON THE NECESSITY FOR BI-DIRECTIONAL COMMUNICATION

Ellen E. Robertson
Texas Utilities Services Inc.

Dottie Cunningham
Celanese Fibers Inc.

ABSTRACT

As the function of capacity management has evolved and matured, considerable attention has been directed towards what the capacity planner and performance analyst must provide management in terms of reporting, or in support of the corporation's decision making progress. A recent survey of CPE practitioners indicates concerns on their part relating to the information communicated to them by management in support of their activities, and to management's view of the CPE function in general. This panel of capacity planners and performance analysts will present these concerns to the audience and provide a forum for discussion on how issues resulting from these concerns might be resolved.

The following list provides a number of questions and issues that will be addressed by the panel. Attendees are encouraged to expand this list by presenting additional questions or issues to the panel during the course of the session.

1. As a CPE practitioner, how do you view your corporation's commitment to capacity planning and performance evaluation?
   - A. Are you properly positioned within the organization?
   - B. Is sufficient manpower devoted to the function?
   - C. Do you have a clearly defined career path?
   - D. Do you receive adequate tools and training?
   - E. Are you frequently assigned non related tasks and projects such that capacity planning and/or performance is really only a 'part time' job?
   - F. Is management willing to listen to problems you have doing your job?
2. Do you receive all the information necessary for you to perform your job properly?
   - A. Are management objectives and expectations clearly defined and communicated to you?
   - B. If there is a corporate strategic plan, do you receive it?
   - C. Do you have access to budget information that may impact the studies or recommendations you are asked to perform?
   - D. Are plans regarding new systems or products provided?
   - E. Do you have ready access to all levels of management across organizational lines?
3. Regarding the information you do receive:
   - A. Is it timely?
   - B. Is it pertinent?
   - C. What is most useful?
   - D. What is least useful?
4. What single piece of information which you do not currently receive, or do not receive on a timely basis, would be most useful to you in performing your job?
5. How do you view your ability to influence corporate plans?
   - A. Do you or your group participate in the formulation of strategic plans?
   - B. Are you free to make recommendations that may not agree with current policy or trends within your company?
   - C. Is the time and effort required to address speculative issues weighed against the potential benefits?
   - D. Are your recommendations generally accepted, and if not is the rational behind their rejection communicated to you?
   - E. Is the resulting net value to the company of actions taken based on your recommendations quantified?
6. What are the three major obstacles you encounter in performing your job?
7. What do you perceive as the most important communication problem between capacity planners/ performance analysts and executives?
8. If you could make one single request of your management relating directly to your job, what would it be?

Executive Views and Expectations of Capacity Planning

James A. Stanton
Morino Associates
Vienna, VA 22180

Abstract: Surveys of capacity management organizations have shown that some of the leading problems experienced by capacity planners are management related. Examples of problem areas include obtaining and keeping management credibility, communicating plans to management, and management support in terms of tools, education, etc. This session will provide a valuable opportunity for CPE practitioners to directly address questions of interest to a panel of executive level managers.

The following list provides a number of the questions and issues that will be addressed by the panel. Attendees are welcomed to expand this list by presenting additional questions or issues to the panel during the course of the session.

1. As a senior executive manager, how do you view the capacity planning (CP) function in your enterprise?
   a. What is the purpose of capacity planning in your enterprise?
   b. What do you think the purpose should be?
   c. What are the objectives of capacity planning?
   d. What is the importance of capacity planning?
   e. Who are the executive participants in the capacity planning process?
   f. What is the role of executive management in the capacity planning process?
   g. What role should the users play in the capacity planning process?

2. Does your organization have a formal CP program?
   a. Where does CP report in the organization?
   b. Where should it report in the organization?
   c. How long has the function been in existence?
   d. How many people are dedicated to the capacity planning process?
   e. What percentage of your DP budget is allocated to CP?

3. What are the three major obstacles to CP in your organization?

4. What formal reports does CP provide management? What should they provide? How often should the reports be provided?

5. Does your CP group participate in preparing the reports listed below? If so, how effective is this participation for the:
   a. Five year strategic plan?
   b. Two year business plan?
   c. Annual operating report?
   d. Quarterly assessment report?

6. How does your CP group translate CP methodology into reports presentable to I/S and executive management?
   a. What should the audit trail look like for CP recommendations?
   b. What kind of supporting information do you find essential in providing credibility to capacity plans?

7. How are user and corporate business forecasts used in your capacity planning process?

8. What do you expect from your CP function and staff?
   a. Why may it not be living up to your expectations?
   b. How is credibility best won and maintained by the capacity planners?
   c. What destroys the credibility of the capacity planners the quickest?
   d. What is the most important communication problem between executives and capacity planners?
   e. What does the executive manager typically want to know from the capacity planning process?
   f. What is the best form of presentation of capacity planning information?
   g. What skills does the executive look for in capacity planners?

9. How does your enterprise handle the management of the CP function?
   a. Should the executives explain their decisions or lack of capacity plan decisions to the capacity planners?
   b. What kinds of complaints are expressed in the executive suite regarding I/S service and how do the executives relate these complaints to CP issues?

10. How do you recommend the capacity planners relate to executive management concerning their careers and their future in your enterprise?
    a. Do executives want to hear about the problems capacity planners have in doing their jobs?
    b. What is your view of the career paths for capacity planners in your enterprise?
    c. What is the best way for the capacity planner to approach executive management to obtain approvals for training and CP tools?
    d. How would you describe the management support (in terms of tools, education, management involvement, and access to business management information) given to the capacity planning process in your enterprise?

11. What single recommendation would you make to the capacity planners in our audience as to how best to deal with executive management?

a. Capacity plannner should know the business of their key users?
b. Capacity planners must understand the primary business concerns, opportunities, areas of emphasis, and weaknesses?
c. I/S employees must bring opportunities to the attention of management?
d. The financial return of capacity plans should be clearly demonstrated?
e. The financial impact of capacity plan alternatives should always be clearly defined?
f. Capacity plan presentations should begin with statements of assumptions and should provide an opportunity for executives to ask and get answers to "what if the assumptions were different", types of questions?

SUCCESSFUL SOFTWARE PERFORMANCE ENGINEERING

MODERATOR: DR. CONNIE U. SMITH

At CMG86, a BOF session reviewed the evolution of Software Performance Engineering (SPE) and discussed recent experiences in applying it. SPE advocates building performance into new systems rather than tuning code after implementation. The BOF session revealed that there are, unfortunately, still problems in convincing managers and software developers of its vital importance. Once convinced, there are still many pitfalls that threaten successful SPE application.

This panel has two parts: the first addresses problems in convincing organizations to abandon the "fix it later" approach and adopt performance engineering. It begins with an overview of what it is, how it helps software development projects and capacity planning, who else is doing it, etc. Then panel members discuss SPE adoption considerations. The second part is on SPE experiences. Panel members will relay their project experiences, discuss pitfalls and offer advice for avoiding them, and highlight successful strategies and techniques. Both managers and practitioners are represented. Time is allocated for answering audience questions.

SPE DEFINITION

Software Performance Engineering (SPE) is a method for constructing software systems to meet responsiveness goals. With it, one models software requirements and designs, and evaluates whether predicted performance metrics meet the specified goals. If not, alternatives are proposed and assessed. The process continues through the detailed design, coding, and testing stages to develop more precise models of the software and its predicted performance.

SPE focuses on software development, but it is also an important part of a capacity management program. There are three major capacity management considerations: planning, growth, and control. Accurate planning requires data on new applications and data on the effect on resource requirements of growth in current work; SPE models provide both.

A passive capacity planning program anticipates growth and reacts to it. SPE advocates an *active* capacity management program that controls configuration requirements. Growth in workload demands may be inevitable, but growth due to resource requirements of software can be controlled through SPE.

SESSION 1:

SPE ADOPTION CONSIDERATIONS

Dr. Thomas E. Bell, Rivendel Consultants:

THE CHALLENGE OF PERFORMANCE ENGINEERING IS DESIGN IMPROVEMENT

Is Performance Engineering an analytical effort to expose the inadequacies of designers and implementors? Is it primarily a control effort involving reviews of designs so that errors can be caught and designers can be ordered to restart their efforts? Does it just involve projecting the amount of resources that will be required to support a new application -- but projecting it as early as possible? It is none of these. Instead it is an effort integral to the development project that contributes in the same way that database design contributes. With an expert understanding of the performance effects of various design alternatives, and the knowledge of how to quantify these effects, the Performance Engineer can suggest alternatives to the rest of the team. He or she can quantify the trade-offs and justify alternatives that, even if less elegant, are necessary to attain the project goals.

The obvious responsibilities of the Performance Engineer -- early projection, estimating equipment requirements, detection of performance bugs -- are less important than *getting* the performance requirements correct, *improving* the design, and *suggesting* improved implementations. Of course, the primary responsibilities are more difficult than the obvious ones; they require Performance Engineers to work cooperatively with other designers. They must use their design abilities, persuasive skills, and quantitative understanding rather than arrogance and power-plays to win over designers whose emphasis is on improved functionality or improved elegance.

Ken Kolence, Kolence Associates:

SOFTWARE PERFORMANCE ENGINEERING IS A JOINT EFFORT

There are several problems that hinder adoption of SPE:

1) it is difficult to quantify its value -- success is the absence of performance problems while only failures are obvious;

2) performance requirements for new systems are seldom explicit;

3) more products are needed to automate the difficult SPE tasks and better integrate SPE with other well-established development activities.

There are many ways to improve the situation. SPE should be more tightly coupled with capacity planning activities. We should be concerned not only with performance but also with workload characterization and forecasting. Success in these activities is easier to quantify. We also need to integrate instrumentation directly into software designs. This provides data for predicting response times under varying loads. It could also feed analytic models for determining required capacity allocation (setting SRM and performance group parameters. The ideal place for such features is within new CASE tools, but it is difficult to convince CASE developers to do it. We must first prove its value.

We must change the viewpoint of software developers. They need to be accountable for ***all*** requirements, including explicit performance goals. Often systems analysts are not aware of reasonable service level objectives, and developers are unaware of techniques for achieving them. We need to develop standards for goal specification. Products facilitate these tasks.

Many SPE tasks are labor intensive. It is difficult to obtain the data needed for SPE studies. Many organizations now use code generation tools and tools that add structure to old "spider web" programs. These products could be easily augmented to provide SPE data.

Finally, we need to generate more interest outside of CMG. In particular, we need to increase awareness of managers, developers, and software engineering researchers.

Ralph Terkowitz, Washington Post:

SOFTWARE PERFORMANCE AT THE WASHINGTON POST

The IBM computer complex at *The Washington Post* has about 20 MIPs of installed processing power with 30 Gigabytes of DASD. We are very proud of the performance of our online systems. Despite a relatively high transaction volume per installed MIP, we average completion of 85 - 90% of all tasks in under one second (compared to a Service Level Commitment of 75%). The unusual part of our environment is that we have accomplished these goals without any full-time analysts!

This juxtaposition of systems performance with lack of performance analysts may seem unusual; however, I believe it is directly related to our adopting Software Performance Engineering (SPE) techniques about five years ago. No one person on our staff is charged with the performance of our systems; rather, each of our senior analysts is responsible for the performance of his own system. Performance requirements and systems performance forecasts are routinely developed and confirmed by the applications staff. Why: Because they understand that developing a system to meet performance criteria is as important as having the system meet any other functional criteria.

The unusual empahsis on applications performance at The Post is not the result of any management revelation. In fact, senior management has no idea that our record of performance-oriented system implementation is not common in the data processing industry. Proposals have never been made to senior management to adopt the SPE technology. They don't even know what it is. SPE has become our method for applications development because of the successes we have had with it.

People often ask us for tips on starting or gaining management support for a SPE program. I believe that there is no one right way: there is nothing magic to do or tell senior management. I think that the key to developing a Software Performance Engineering program is recognizing that SPE is not in itself the goal of your development; rather, it is the means to the end of developing high performance systems -- *systems that consume no more than the forecast resources and deliver the expected performance.*

SPE is now a way of life at The Post. It is not in place due to management fiat, but rather because we have shown senior management that we can forecast, develop and build systems that operate at the expected cost with the expected performance. Management doesn't care about the method; they care about the results from it!

Dr. Connie U. Smith, L&S Computer Technology, Inc.:

STRUCTURED PROGRAMMING, HYBRID CORN, AND PERFORMANCE ENGINEERING

On the surface these appear to have little in common. However, recent software engineering studies have examined the process of technology transfer to understand what happens and to see if it can be hastened. They have found that it typically takes 15-20 years between the time that new technologies emerge and their widespread use in industry. The range holds for most new technologies whether they are software related, such as structured programming, or in other domains, such as hybrid corn. SPE is approximately 10 years old. Will it be commonplace in the next 5-10 years? Can the time be shortened? According to technology transfer research, you can play a role in its adoption.

Technologies are more readily adopted when they have a strong advocate who: persistently encourages and follows up on its use, serves as a two-way communication channel between technology users and technology developers, and helps to resolve problems that arise. Technologies are more likely to be adopted when they have a clear benefit; they are compatible with existing methods, experiences, and needs; they are easy to understand; easy to try; and they produce a measurable result. When an organization tries a new technology, it is more likely to be successful if an expert participates in the project and contributes to its success. These needs must be addressed if SPE is to be adopted and used. So, if you believe in SPE concepts and want to see them used in your organization:

- Learn what it is so you can explain it and its benefits to others
- Find or become a persistent advocate
- Document compatibility in your environment
- Apply it to a pilot project, then explain it and its results to others
- Quantify the results of using SPE (otherwise success is invisible)
- Alternatively, point out problems that occur without SPE and ***how*** they can easily be prevented.

Document experiences so others can point to its successful uses for justification, and learn from problems so they can be avoided.

SESSION 2:

SPE EXPERIENCES

Greg Fox, TRW Federal Systems Group:

TAKE PRACTICAL PERFORMANCE ENGINEERING STEPS EARLY

The following remarks address performance engineering in the context of a "large" system development. "Large" means hundreds of thousands of lines of software designed, coded, and tested by dozens of developers in a multi-year effort. The system performs complex transactions, with extensive algorithmic processing, on streams of data or messages received from sources external to the system.

The success of a performance engineering effort can be gauged by how well the finished system meets performance requirements ***and*** by how much disruption there was to the normal development process. Many complex, modern systems require some disruption; the goal is to keep it to acceptable levels.

Performance engineering has two phases: theoretical and practical. The success of the effort is determined by how and when the transition from theoretical to practical occurs. The theoretical aspects (analysis, modeling, and simulation) are valuable in the development of algorithms, an architecture free of unrecoverable bottlenecks, and realistic estimates of hardware resource requirements. The practical phase incorporates the performance features into the software. During integration it is unrealistic to expect that measured system performance will be as good as predicted by the models. The models have inherent limitations in accounting for the following factors: 1) the system environment, including the performance aspects of complex operating systems, and 2) inaccuracy of resource consumption estimates, including the effect of varying skill levels among programmers.

Practical performance engineering steps during design, code, and unit test narrow the gap between model predictions and initial measured performance. This reduces the amount of improvement needed during integration. By the time top-level design is completed, emphasis should shift from analysis and prediction to more practical activities, such as developing information for programmers to build the system in a performance-efficient fashion. A benchmark design and measurement activity should parallel coding and unit testing, and be an integral part of the development by the time integration is underway.

Experience shows that for many large systems, a factor of two performance improvement can be achieved from early integration benchmarks with relatively little disruption of the integration process. An additional improvement factor of two to three may be possible with moderate disruptions to cost and schedule. If measurement is left until late in integration, required performance improvements can cause major disruption to the completion of the development effort.

Norb Trohoski, General Electric:

ACCEPTANCE IS THE KEY ASPECT OF EFFECTIVE PERFORMANCE ENGINEERING

The key aspect of successful performance engineering in large computer system developments, where there are many pieces of software being developed involving hundreds of engineers and programmers, is its acceptance by both engineering and software development. This acceptance and commitment produces the proper team effort required to build in the performance.

The most common design and development methodology has been to develop the software programs so that they get the right results without concern for performance issues. Programmers do not intentionally write poor performing code, rather, runtime performance is not a major consideration in the design. Not until the programmer is in trouble with runtime does he become interested in performance analysis. This generally results in cost overruns, missed delivery schedules, and upset upper management and customers. Even when a performance engineering activity is in place, if the engineers and programmers do not believe in it, the information they provide on the proposed design is usually late, sketchy, and incomplete. This is because they are required to produce the data and, since they do not believe in performance engineering, they do as little as they can get away with and still meet their commitment. The worst part is when the data is incomplete. This may produce an analysis of the software that indicates the software will meet performance requirements, and then it is not discovered until test that the software will not.

If the developers give the performance engineering approach a try usually it will result in believers. Those who have experienced what performance engineering can do for them continually ask the performance engineering group for support. This makes the performance engineering activity easy, with resulting systems that are capable of meeting their performance requirements.

Thus it has been found that the key aspect to effective Performance Engineering is to get engineers and programmers to understand and believe in the importance of Performance Engineering; when this happens, successful software systems are produced.

Doug McBride, Hewlett Packard:

APPROPRIATE DECOMPOSITION AND MEASUREMENT ARE VITAL

There are, of course, a myriad of steps in software performance engineering. I would say, however, the two most troublesome areas for designers and implementors are:

- Decomposing the software system into components for subsystem performance measurement and tuning.

- Locating and/or creating software or hardware performance monitoring tools which give an accurate picture of how the software component is performing,and what system resources it is consuming.

As in solving any complex problem, decomposing the larger software system into components by breaking it into smaller parts typically aids in the solution of the overall problem. Decomposition focuses attention and resources on a smaller, more manageable subset of the entire problem. Three areas to consider when performing the decomposition include:

- Delineating component boundaries based on system resource consumption characteristics, as well as anticipated sensitivity to overall system performance parameters.

- Establishing meaningful performance budgets and goals for the particular component according to its functional relationship to the other components, and to overall system performance.

- Having measurement tools in place, or under development, to facilitate proper monitoring of the progress of the components being tested.

Without the tools that provide a window into the system to determine how well individual software components are performing and using system resources, it is virtually impossible to carry out the performance engineering function with any possibility of quantifiable success. Many ambitious performance efforts have failed due to the lack of proper monitoring tools - software or hardware - needed to meet performance goals and validate resource budgets. It is vital that these tools, which can take many forms, depending on prior experience and current need, be available, or their creation be budgeted during the design phase of the software. It is equally vital to explain and discuss the types of tools needed with project management, and how and when to acquire them.

PROCEEDINGS SUBJECT INDEX

Accounting

A Simple Linear Programming Model For Computation Center Investment Decisions, **Merges,** 64.
Transaction Resource Demand, **Cheeseman,** 345.
Capacity Budgets And Financial Budgets: Relating Them To Each Other Through Resource Accounting, **Richardson,** 538.
Experience in Cost Allocation, **Stancavish,** 543.
Chargeback vs Cost Accounting: The Appropriate Emphasis, **Gillen,** 546.
Can Service Units Be Used For Uniform Chargebacks?, **Claunch,** 617.
A Strategy For Improving IO Subsystem Design Using Cost Per IO Analysis, **Jones,** 672.

Application Development

The System Performance Cycle: A Methodology For Architecting Product Performance, **McBride,** 145.
Expert Systems: Let's Build One Together, **Ingrassia,** 151.
A Flexible Approach To Performance Prediction For Large System Software Development, **Long,** 167.
Sizing New Automated Systems Having Unknown Workloads: A Structured Approach, **Gibson,** 177.
Experiences With Performance Engineering, **Bell,** 549.
A Systems Engineering Look at SAA, **Cook,** 636.

Benchmarking

Projection Algorithms for IBM DASD Cache Analysis Tool, **Ju,** 1.
A Comparative Study of System/370 CPU's, **Houtekamer,** 72.
Modelling Memory Constrained Systems, **Houtekamer,** 102.
Performance Comparison Of Vector Processors On Large Scientific and Business Machines, **Lindsay,** 310.
3480 Performance and Capacity Planning, **Keller,** 336.
A Benchmark Comparison of DB2 and the DBC/1012, **Steindel,** 360.
Characterizing UNIX Stream Mechanism Via Pipe Experiments, **Lee,** 452.
Real World Performance Measurements For Unix Based Systems, **Nelson,** 574.
High Performance Processor Design, **McCormack,** 736.
P. S. I Love You. Signed TSO, **Lichtig,** 835.
A Survey of Performance Applications of The Teleprocessing Network Simulator (TPNS), **Burke,** 880.

CDC Systems

Estimating The Maximum Throughput Of A Cyber 930 Disk Subsystem, **Sutherland,** 304.

CICS

Identification Of Tuning Opportunities Through The Use Of User Behavior Graphs, **Hackett,** 33.
VSAM Specification and Tuning: Optimization of VSAM Performance in Batch and CICS Environments, **Bereznay,** 40.
New Measurement Data Sources For MVS/XA and Its Subsystems, **Merrill,** 222.
A Heurustic Modeling Approach To CICS Capacity Planning, **Hackenberg,** 252.
A Systematic Approach For Identifying Peak Workload Intervals, **Artis,** 287.
Capacity Planning For A Multi-Processor Environment Supporting CICS Transactions, **Gantz,** 405.
CICS Performance and Workload Reports, **Comstock,** 767.
CICS and DB2: A Marriage Made in Heaven?, **Shane,** 822.
Modelling CICS Systems: The Performance Impact Of MRO Implementations And Data Base Considerations, **Friedman,** 828.
A Survey of Performance Applications of The Teleprocessing Network Simulator (TPNS), **Burke,** 880.
CICS - Designing For Performance, **Emanuel,** 970.

CPE Studies

Identification Of Tuning Opportunities Through The Use Of User Behavior Graphs, **Hackett,** 33.
VSAM Specification and Tuning: Optimization of VSAM Performance in Batch and CICS Environments, **Bereznay,** 40.
In Pursuit of DASD Data Path Alternatives, **Duhl,** 46.

An Intuitive Approach To Performance Prediction with Application to Workload Management in VM SP/HPO, **Hellerstein,** 53.
A Comparative Study of System/370 CPU's, **Houtekamer,** 72.
Modelling Memory Constrained Systems, **Houtekamer,** 102.
The System Performance Cycle: A Methodology For Architecting Product Performance, **McBride,** 145.
Expert Systems: Let's Build One Together, **Ingrassia,** 151.
The Use Of Regression Analysis In The Calculation Of Capture Ratios, **Kirshin,** 189.
Cacheing the Tape Library—Tape Reference Patterns and Automated Tape Systems, **Kaberon,** 192.
Analyzing The Performance Characteristics Of 18-Track Tape Technology, **Dines,** 201.
Static vs. Dynamic Management of Consistently Very Active Data Sets, **Henley,** 208.
Expert Systems And MVS Performance Management, **Van Kinsbergen,** 233.
Expanded Storage Usage And Tuning, **Hanna,** 237.
Measurement and Analysis of Expanded Storage in IBM 3090-400 Systems, **Buzen,** 246.
Paging Analysis In An Expanded Storage Environment, **Beretvas,** 256.
IBM 3090 Processor Expanded Storage Experiments, **Pilch,** 266.
A Systematic Approach For Identifying Peak Workload Intervals, **Artis,** 287.
Workload-Oriented Performance Analysis, **Samson,** 290.
On The Optimization of Memory in Large Interactive Environments, **Mungal,** 296.
Estimating The Maximum Throughput Of A Cyber 930 Disk Subsystem, **Sutherland,** 304.
Performance Comparison Of Vector Processors On Large Scientific and Business Machines, **Lindsay,** 310.
A Technique For Evaluating The Performance Of Vector Processors, **Lindsay,** 330.
3480 Performance and Capacity Planning, **Keller,** 336.
Performance Tuning For A VAX-Based Relational Database Application: A Case Study And Lessons Learned, **Allison,** 370.
Large Capacity DASD: Is Performance Something To Be Afraid Of?, **McNutt,** 399.
A Method For Reporting Cached I/O SubSystem Performance, **Nearing,** 432.
Sensitivity Analysis Of System Changes Under A Production Environment, **Lo,** 446.
Characterizing UNIX Stream Mechanism Via Pipe Experiments, **Lee,** 452.
A Performance/Capacity Data Base For Long Term Trends And Studies, **Jack,** 522.
Experiences With Performance Engineering, **Bell,** 549.
Service Level Agreements "Good Fences Make Good Neighbors", **Miller,** 553.
Capacity Management For Laser Optical Devices, **Sarna,** 591.
Magnetic Cartridge Tape Configuration Management, **Guerrero,** 596.
Incorporating Data Networks Into A Performance And Capacity Planning Structure — A Case Study, **Smith,** 606.
Systems Performance Management and Capacity Planning, **Domanski,** 725.
High Performance Processor Design, **McCormack,** 736.
The Theory And Practice Of I/O Tuning, **Olcott,** 749.
Online Data Base vs. Raw System Data, **Denis,** 761.
VM/HPO Performance Measurements, **Streger,** 795.
Accounting for Workload Variation in Interval Selection and Analysis, **Williams,** 808.
Tools And Methods Available For In-Depth VM Performance Analysis, **Christopher,** 816.
I/O Tuning In The MVS/XA Environment, **Olcott,** 889.
Paging And Swapping Interactions, **Buzen,** 929.
Performance Evaluation Of Future System Designs, **Quernemoen,** 936.
3090-Class Processor Architectures and Workload Performance Levels, **Keeney,** 947.
Coupled Systems For Performance, **Cohen,** 949.
A Computer and Communications Network Performance Analysis Primer, **Stuck,** 959.
A Close Look At MVS Systems: Mechanisms, Performance And Security, **Paans,** 962.
CICS - Designing For Performance, **Emanuel,** 970.
Rules of Thumb: Science or Science Fiction, **Zimmer,** 981.

Capacity Planning

A Comparative Study of System/370 CPU's, **Houtekamer,** 72.
Tightly-Coupled Effect Processor Contention Overhead, **Place,** 79.
Forecasting CPU Utilization in an IMS Environment: A Case Study Using Monte Carlo Techniques, **Gray,** 81.
Modelling Of A Production IMS System Under A Multi-Function ProcessingArchitecture, **Gilbert,** 87.
Memory Management For Large IBM Systems, **Lee,** 95.
PROFS Capacity Planning, **Shub,** 110.
VM Performance - Top Down That Went Bottom Up, **Dickerson,** 118.

Expert Systems: Let's Build One Together, **Ingrassia,** 151.
A Flexible Approach To Performance Prediction For Large System Software Development, **Long,** 167.
Sizing New Automated Systems Having Unknown Workloads: A Structured Approach, **Gibson,** 177.
An Experiment in the Calculation of Capture Ratios, **Temperly,** 184.
The Use Of Regression Analysis In The Calculation Of Capture Ratios, **Kirshin,** 189.
A Simulation Model for Determining Optimal Freespace Levels on MVS Disk Storage Volumes, **Levy,** 217.
Expanded Storage Usage And Tuning, **Hanna,** 237.
A Heurustic Modeling Approach To CICS Capacity Planning, **Hackenberg,** 252.
Is A Picture Always Worth A Thousand Words?, **MacKinnon,** 272.
An Implementation Of A Graphical Management Information System Using Symphony, **Wilson,** 279.
A Systematic Approach For Identifying Peak Workload Intervals, **Artis,** 287.
Estimating The Maximum Throughput Of A Cyber 930 Disk Subsystem, **Sutherland,** 304.
Performance Comparison Of Vector Processors On Large Scientific and Business Machines, **Lindsay,** 310.
A Technique For Evaluating The Performance Of Vector Processors, **Lindsay,** 330.
Workload Characterization Using UNIX System Accounting, **Tornheim,** 351.
Capacity Planning For A Multi-Processor Environment Supporting CICS Transactions, **Gantz,** 405.
Model Timeframe Selection And Analysis Of Variance Of An Online Transaction Processing Workload, **Henry,** 415.
Balanced Exponential Approximation Algorithm, **Kumar,** 441.
Sensitivity Analysis Of System Changes Under A Production Environment, **Lo,** 446.
Capacity Analysis Of PBX Systems, **Lo,** 464.
A Comprehensive Process For Projecting Computer Resources, **Wandzilak,** 482.
Capacity Planning As A Business Process, **Rosenberg,** 490.
A Capacity Planning Methodology That Is Destined To Succeed, **Lawrence,** 496.
What We Can Learn From Other Capacity Planners: A Survey of Quantitative Methods, **Ingraham,** 504.
Quality Improvement In Capacity Management, **Wessner,** 518.
A Performance/Capacity Data Base For Long Term Trends And Studies, **Jack,** 522.
Financial Considerations In Capacity Planning, **Rosenberg,** 531.
Capacity Budgets And Financial Budgets: Relating Them To Each Other Through Resource Accounting, **Richardson,** 538.
Capacity Planning in the Age of Competitive Advantage Computing, **Lipner,** 561.
Structured MIS Planning, **Salminen,** 566.
Incorporating Data Networks Into A Performance And Capacity Planning Structure — A Case Study, **Smith,** 606.
Some Experiences of Data Centre Relocation, **Day,** 608.
The Role Of Capacity Planning In Consolidations, Acquisitions and Mergers, **Verbil,** 611.
Establishing Useful Business Relationships, **Allen,** 630.
DASD Management: Let's Talk Money, **Parker,** 651.
An Introduction to Business Element Forecasting, **Bowerman,** 703.
A Practical Use Of Natural Forecasting Units, **Reyland,** 710.
After The Study Is Over, **Cooley,** 714.
Systems Performance Management and Capacity Planning, **Domanski,** 725.
Fundamentals Of Statistical Workload Forecasting, **Artis,** 733.
A Capacity Planning/Queueing Theory Primer, **Bolker,** 734.
Implementing Network Capacity Planning, **Olberding,** 741.
Capacity Planning By The KISS Method, **Kayser,** 797.
P. S. I Love You. Signed TSO, **Lichtig,** 835.
A Capacity Planning Methodology For Data Networks, **Harmon,** 858.
From Simple Regression To Analysis Of Variance And Beyond—Painlessly, **Pool,** 901.
3090-Class Processor Architectures and Workload Performance Levels, **Keeney,** 947.
A Close Look At MVS Systems: Mechanisms, Performance And Security, **Paans,** 962.
Good Tools And Models Are Not Enough: On The Necessity For Bi-Directional Communication, **Robertson,** 988.

DASD

Projection Algorithms for IBM DASD Cache Analysis Tool, **Ju,** 1.
A Multiple-Workload Approach to Cache Planning, **McNutt,** 9.
VSAM Specification and Tuning: Optimization of VSAM Performance in Batch and CICS Environments, **Bereznay,** 40.
In Pursuit of DASD Data Path Alternatives, **Duhl,** 46.
Static vs. Dynamic Management of Consistently Very Active Data Sets, **Henley,** 208.
A Simulation Model for Determining Optimal Freespace Levels on MVS Disk Storage Volumes, **Levy,** 217.

Estimating The Maximum Throughput Of A Cyber 930 Disk Subsystem, **Sutherland,** 304.
Empirical Queueing Models of DASD Response Time, **Major,** 390.
Large Capacity DASD: Is Performance Something To Be Afraid Of?, **McNutt,** 399.
Disk Cache Performance Modeling, **Salsburg,** 423.
A Method For Reporting Cached I/O SubSystem Performance, **Nearing,** 432.
Analyzing File System Fragmentation on UNIX Systems, **Samadi,** 458.
Storage Performance, An Eight Year Outlook, **Goldstein,** 579.
Cache Management in a Hostile Environment, **Wells,** 586.
DASD Management: Let's Talk Money, **Parker,** 651.
Report On The Large System Storage Hierarchy, **Percell,** 685.
The Theory And Practice Of I/O Tuning, **Olcott,** 749.
High Performance I/O Subsystems, **Mungal,** 916.

DEC Systems

Performance Tuning For A VAX-Based Relational Database Application: A Case Study And Lessons Learned, **Allison,** 370.
Conceptual Issues In Measuring The Behavior Of A Distributed Real-Time Process-Control Computer System, **Lawrence,** 474.

Database Systems

Workload Characterization for Subsystems: A Case Study for IDMS, **Friedman,** 16.
Modelling Of A Production IMS System Under A Multi-Function ProcessingArchitecture, **Gilbert,** 87.
New Measurement Data Sources For MVS/XA and Its Subsystems, **Merrill,** 222.
A Benchmark Comparison of DB2 and the DBC/1012, **Steindel,** 360.
Performance Tuning For A VAX-Based Relational Database Application: A Case Study And Lessons Learned, **Allison,** 370.
CICS and DB2: A Marriage Made in Heaven?, **Shane,** 822.

Expert Systems

Expert Systems: Let's Build One Together, **Ingrassia,** 151.
A PROLOG-based Expert System For Tuning MVS/XA, **Domanski,** 160.
Expert Systems And MVS Performance Management, **Van Kinsbergen,** 233.
A Fully Automatic Analytic Approach to Budget-constrained System Upgrade, **Chanson,** 620.
Incorporating Expert Systems Technology Into Software Performance Engineering, **Weishar,** 720.

I/O Systems

Projection Algorithms for IBM DASD Cache Analysis Tool, **Ju,** 1.
A Multiple-Workload Approach to Cache Planning, **McNutt,** 9.
VSAM Specification and Tuning: Optimization of VSAM Performance in Batch and CICS Environments, **Bereznay,** 40.
In Pursuit of DASD Data Path Alternatives, **Duhl,** 46.
Memory Management For Large IBM Systems, **Lee,** 95.
VM Performance - Top Down That Went Bottom Up, **Dickerson,** 118.
Cacheing the Tape Library—Tape Reference Patterns and Automated Tape Systems, **Kaberon,** 192.
Analyzing The Performance Characteristics Of 18-Track Tape Technology, **Dines,** 201.
Static vs. Dynamic Management of Consistently Very Active Data Sets, **Henley,** 208.
Estimating The Maximum Throughput Of A Cyber 930 Disk Subsystem, **Sutherland,** 304.
3480 Performance and Capacity Planning, **Keller,** 336.
A Benchmark Comparison of DB2 and the DBC/1012, **Steindel,** 360.
Empirical Queueing Models of DASD Response Time, **Major,** 390.
Large Capacity DASD: Is Performance Something To Be Afraid Of?, **McNutt,** 399.
Disk Cache Performance Modeling, **Salsburg,** 423.
A Method For Reporting Cached I/O SubSystem Performance, **Nearing,** 432.
Storage Performance, An Eight Year Outlook, **Goldstein,** 579.
Cache Management in a Hostile Environment, **Wells,** 586.
Capacity Management For Laser Optical Devices, **Sarna,** 591.
Magnetic Cartridge Tape Configuration Management, **Guerrero,** 596.
A Fully Automatic Analytic Approach to Budget-constrained System Upgrade, **Chanson,** 620.
A Strategy For Improving IO Subsystem Design Using Cost Per IO Analysis, **Jones,** 672.

Report On The Large System Storage Hierarchy, **Percell,** 685.
The Theory And Practice Of I/O Tuning, **Olcott,** 749.
I/O Tuning In The MVS/XA Environment, **Olcott,** 889.
High Performance I/O Subsystems, **Mungal,** 916.
A Close Look At MVS Systems: Mechanisms, Performance And Security, **Paans,** 962.

IMS

Forecasting CPU Utilization in an IMS Environment: A Case Study Using Monte Carlo Techniques, **Gray,** 81.
Expert Systems: Let's Build One Together, **Ingrassia,** 151.
Transaction Resource Demand, **Cheeseman,** 345.

Installation Management

A Simple Linear Programming Model For Computation Center Investment Decisions, **Merges,** 64.
End-User Response Time Measurement - Not The Final Solution, **Futtrup,** 122.
Determining End-User Response Time Using NLDM, **Yount,** 133.
General Principles for Performance Oriented Design, **Smith,** 138.
Cacheing the Tape Library—Tape Reference Patterns and Automated Tape Systems, **Kaberon,** 192.
Static vs. Dynamic Management of Consistently Very Active Data Sets, **Henley,** 208.
Is A Picture Always Worth A Thousand Words?, **MacKinnon,** 272.
Transaction Resource Demand, **Cheeseman,** 345.
A Comprehensive Process For Projecting Computer Resources, **Wandzilak,**482.
Capacity Planning As A Business Process, **Rosenberg,** 490.
A Capacity Planning Methodology That Is Destined To Succeed, **Lawrence,** 496.
What We Can Learn From Other Capacity Planners: A Survey of Quantitative Methods, **Ingraham,** 504.
Managing Departmental Mainframe Demand Through Shared Accountability, **Jones,** 508.
Management Reporting: An Approach To Availability And Service Reporting, **Marcus,** 515.
Quality Improvement In Capacity Management, **Wessner,** 518.
A Performance/Capacity Data Base For Long Term Trends And Studies, **Jack,** 522.
Financial Considerations In Capacity Planning, **Rosenberg,** 531.
Capacity Budgets And Financial Budgets: Relating Them To Each Other Through Resource Accounting, **Richardson,** 538.
Experience in Cost Allocation, **Stancavish,** 543.
Chargeback vs Cost Accounting: The Appropriate Emphasis, **Gillen,** 546.
Experiences With Performance Engineering, **Bell,** 549.
Service Level Agreements "Good Fences Make Good Neighbors", **Miller,** 553.
Capacity Planning in the Age of Competitive Advantage Computing, **Lipner,** 561.
Structured MIS Planning, **Salminen,** 566.
Who Goes There? A Dialogue Of Questions And Answers About Benign Hacking, **Stevens,** 569.
Cache Management in a Hostile Environment, **Wells,** 586.
Capacity Management For Laser Optical Devices, **Sarna,** 591.
Magnetic Cartridge Tape Configuration Management, **Guerrero,** 596.
End-User Performance Measurements: An Automated PC Approach, **Martin Flickinger,** 599.
Some Experiences of Data Centre Relocation, **Day,** 608.
The Role Of Capacity Planning In Consolidations, Acquisitions and Mergers, **Verbil,** 611.
Can Service Units Be Used For Uniform Chargebacks?, **Claunch,** 617.
Establishing Useful Business Relationships, **Allen,** 630.
A Systems Engineering Look at SAA, **Cook,** 636.
DASD Management: Let's Talk Money, **Parker,** 651.
Bullmoose Does TA: A Manager's Guide to Transactional Analysis, **Stanton,** 698.
After The Study Is Over, **Cooley,** 714.
Computer Acquisition After The Tax Reform Act Of 1986: Exploring The Opportunities, **Amembal,** 719.
The Goldilocks Syndrome, **Berry,** 723.
Senior Management Expectations of Capacity Planning, **Stanton,** 989.

MVS

Projection Algorithms for IBM DASD Cache Analysis Tool, **Ju,** 1.
Memory Management For Large IBM Systems, **Lee,** 95.

Modelling Memory Constrained Systems, **Houtekamer,** 102.
A PROLOG-based Expert System For Tuning MVS/XA, **Domanski,** 160.
An Experiment in the Calculation of Capture Ratios, **Temperly,** 184.
The Use Of Regression Analysis In The Calculation Of Capture Ratios, **Kirshin,** 189.
New Measurement Data Sources For MVS/XA and Its Subsystems, **Merrill,** 222.
Expert Systems And MVS Performance Management, **Van Kinsbergen,** 233.
Expanded Storage Usage And Tuning, **Hanna,** 237.
Paging Analysis In An Expanded Storage Environment, **Beretvas,** 256.
On The Optimization of Memory in Large Interactive Environments, **Mungal,** 296.
A Technique For Evaluating The Performance Of Vector Processors, **Lindsay,** 330.
Sensitivity Analysis Of System Changes Under A Production Environment, **Lo,** 446.
Can Service Units Be Used For Uniform Chargebacks?, **Claunch,** 617.
A Strategy For Improving IO Subsystem Design Using Cost Per IO Analysis, **Jones,** 672.
Report On The Large System Storage Hierarchy, **Percell,** 685.
Systems Performance Management and Capacity Planning, **Domanski,** 725.
Fundamentals Of Statistical Workload Forecasting, **Artis,** 733.
The Theory And Practice Of I/O Tuning, **Olcott,** 749.
P. S. I Love You. Signed TSO, **Lichtig,** 835.
I/O Tuning In The MVS/XA Environment, **Olcott,** 889.
High Performance I/O Subsystems, **Mungal,** 916.
Paging And Swapping Interactions, **Buzen,** 929.
A Close Look At MVS Systems: Mechanisms, Performance And Security, **Paans,** 962.

Management Reporting

Is A Picture Always Worth A Thousand Words?, **MacKinnon,** 272.
An Implementation Of A Graphical Management Information System Using Symphony, **Wilson,** 279.
A Method For Reporting Cached I/O SubSystem Performance, **Nearing,** 432.
A Comprehensive Process For Projecting Computer Resources, **Wandzilak,** 482.
Capacity Planning As A Business Process, **Rosenberg,** 490.
Managing Departmental Mainframe Demand Through Shared Accountability, **Jones,** 508.
Management Reporting: An Approach To Availability And Service Reporting, **Marcus,** 515.
Financial Considerations In Capacity Planning, **Rosenberg,** 531.
Capacity Budgets And Financial Budgets: Relating Them To Each Other Through Resource Accounting, **Richardson,** 538.
Experience in Cost Allocation, **Stancavish,** 543.
Chargeback vs Cost Accounting: The Appropriate Emphasis, **Gillen,** 546.
Some Experiences of Data Centre Relocation, **Day,** 608.
The Role Of Capacity Planning In Consolidations, Acquisitions and Mergers, **Verbil,** 611.
DASD Management: Let's Talk Money, **Parker,** 651.
After The Study Is Over, **Cooley,** 714.
Fundamentals Of Statistical Workload Forecasting, **Artis,** 733.
CICS Performance and Workload Reports, **Comstock,** 767.
Capacity Planning By The KISS Method, **Kayser,** 797.
Reporting Availability At The Job Step Level, **Scott,** 801.

Networks

End-User Response Time Measurement - Not The Final Solution, **Futtrup,** 122.
Determining End-User Response Time Using NLDM, **Yount,** 133.
The Need for a Network Monitor, **Bodley,** 282.
Capacity Analysis Of PBX Systems, **Lo,** 464.
Conceptual Issues In Measuring The Behavior Of A Distributed Real-Time Process-Control Computer System, **Lawrence,** 474.
Management Reporting: An Approach To Availability And Service Reporting, **Marcus,** 515.
End-User Performance Measurements: An Automated PC Approach, **Martin Flickinger,** 599.
Incorporating Data Networks Into A Performance And Capacity Planning Structure — A Case Study, **Smith,** 606.
A Systems Engineering Look at SAA, **Cook,** 636.
Implementing Network Capacity Planning, **Olberding,** 741.
An Introduction To Program-To-Program Communication And Peer-To-Peer Networking In An IBM SNA Environment, **McDonald,** 774.

CICS and DB2: A Marriage Made in Heaven?, **Shane,** 822.
A Capacity Planning Methodology For Data Networks, **Harmon,** 858.
Netview and Netview/PC Tutorial, **Terplan,** 865.
A Survey of Performance Applications of The Teleprocessing Network Simulator (TPNS), **Burke,** 880.
A Computer and Communications Network Performance Analysis Primer, **Stuck,** 959.

Office Automation

PROFS Capacity Planning, **Shub,** 110.
Capacity Analysis Of PBX Systems, **Lo,** 464.

Operations

A Simple Linear Programming Model For Computation Center Investment Decisions, **Merges,** 64.
Tightly-Coupled Effect Processor Contention Overhead, **Place,** 79.
Cacheing the Tape Library—Tape Reference Patterns and Automated Tape Systems, **Kaberon,** 192.

Personal Computers

A PROLOG-based Expert System For Tuning MVS/XA, **Domanski,** 160.
An Implementation Of A Graphical Management Information System Using Symphony, **Wilson,** 279.
End-User Performance Measurements: An Automated PC Approach, **Martin Flickinger,** 599.
Netview and Netview/PC Tutorial, **Terplan,** 865.

Queueing Models

A Multiple-Workload Approach to Cache Planning, **McNutt,** 9.
Tightly-Coupled Effect Processor Contention Overhead, **Place,** 79.
Modelling Of A Production IMS System Under A Multi-Function ProcessingArchitecture, **Gilbert,** 87.
General Principles for Performance Oriented Design, **Smith,** 138.
An Experiment in the Calculation of Capture Ratios, **Temperly,** 184.
Measurement and Analysis of Expanded Storage in IBM 3090-400 Systems, **Buzen,** 246.
A Heurustic Modeling Approach To CICS Capacity Planning, **Hackenberg,** 252.
Workload Characterization Using UNIX System Accounting, **Tornheim,** 351.
Empirical Queueing Models of DASD Response Time, **Major,** 390.
Model Timeframe Selection And Analysis Of Variance Of An Online Transaction Processing Workload, **Henry,** 415.
Balanced Exponential Approximation Algorithm, **Kumar,** 441.
Analyzing File System Fragmentation on UNIX Systems, **Samadi,** 458.
Capacity Analysis Of PBX Systems, **Lo,** 464.
Conceptual Issues In Measuring The Behavior Of A Distributed Real-Time Process-Control Computer System, **Lawrnce,** 474.
What We Can Learn From Other Capacity Planners: A Survey of Quantitative Methods, **Ingraham,** 504.
A Fully Automatic Analytic Approach to Budget-constrained System Upgrade, **Chanson,** 620.
A Capacity Planning/Queueing Theory Primer, **Bolker,** 734.
A Real-Life "How-To-Do-It" Guide To Modelling, **Taylor,** 787.
Analytic Modeling for the Non-Analytic Modeler, **Sannerud,** 790.
What Models Do And What Models Don't, **Levy,** 807.
Modelling CICS Systems: The Performance Impact Of MRO Implementations And Data Base Considerations, **Friedman,** 828.
Good Tools And Models Are Not Enough: On The Necessity For Bi-Directional Communication, **Robertson,** 988.

SNA

End-User Performance Measurements: An Automated PC Approach, **Martin Flickinger,** 599.
A Computer and Communications Network Performance Analysis Primer, **Stuck,** 959.

Simulation Models

A Simulation Model for Determining Optimal Freespace Levels on MVS Disk Storage Volumes, **Levy,** 217.
Disk Cache Performance Modeling, **Salsburg,** 423.
What We Can Learn From Other Capacity Planners: A Survey of Quantitative Methods, **Ingraham,** 504.

Software Engineering

General Principles for Performance Oriented Design, **Smith,** 138.
Experiences With Performance Engineering, **Bell,** 549.
Incorporating Expert Systems Technology Into Software Performance Engineering, **Weishar,** 720.
CICS - Designing For Performance, **Emanuel,** 970.
Successful Software Performance Engineering, **Smith,** 991.

TSO

On The Optimization of Memory in Large Interactive Environments, **Mungal,** 296.
P. S. I Love You. Signed TSO, **Lichtig,** 835.
A Survey of Performance Applications of The Teleprocessing Network Simulator (TPNS), **Burke,** 880.

UNIX

Workload Characterization Using UNIX System Accounting, **Tornheim,** 351.
Characterizing UNIX Stream Mechanism Via Pipe Experiments, **Lee,** 452.
Analyzing File System Fragmentation on UNIX Systems, **Samadi,** 458.
Real World Performance Measurements For Unix Based Systems, **Nelson,** 574.

Univac Systems

The Need for a Network Monitor, **Bodley,** 282.

User Productivity

End-User Response Time Measurement - Not The Final Solution, **Futtrup,** 122.
Determining End-User Response Time Using NLDM, **Yount,** 133.
CICS and DB2: A Marriage Made in Heaven?, **Shane,** 822.

VM Systems

Projection Algorithms for IBM DASD Cache Analysis Tool, **Ju,** 1.
An Intuitive Approach To Performance Prediction with Application to Workload Management in VM SP/HPO, **Hellerstein,** 53.
VM Performance - Top Down That Went Bottom Up, **Dickerson,** 118.
Methodology To Develop A Tool For Monitoring CPU Usage By Applications Running Under VM, **Deese,** 764.
VM/HPO Performance Measurements, **Streger,** 795.
Tools And Methods Available For In-Depth VM Performance Analysis, **Christopher,** 816.
A Survey of Performance Applications of The Teleprocessing Network Simulator (TPNS), **Burke,** 880.

Workload Characterization

Workload Characterization for Subsystems: A Case Study for IDMS, **Friedman,** 16.
Sizing New Automated Systems Having Unknown Workloads: A Structured Approach, **Gibson,** 177.
Workload-Oriented Performance Analysis, **Samson,** 290.
Workload Characterization Using UNIX System Accounting, **Tornheim,** 351.
Model Timeframe Selection And Analysis Of Variance Of An Online Transaction Processing Workload, **Henry,** 415.
Sensitivity Analysis Of System Changes Under A Production Environment, **Lo,** 446.
A Comprehensive Process For Projecting Computer Resources, **Wandzilak,** 482.
A Capacity Planning Methodology That Is Destined To Succeed, **Lawrence,** 496.
Capacity Planning in the Age of Competitive Advantage Computing, **Lipner,** 561.
Structured MIS Planning, **Salminen,** 566.
Workload Characterization And Forecasting For A Large Commercial Environment, **Miller,** 655.
Systems Performance Management and Capacity Planning, **Domanski,** 725.
Fundamentals Of Statistical Workload Forecasting, **Artis,** 733.
Accounting for Workload Variation in Interval Selection and Analysis, **Williams,** 808.

Workload Forecasting

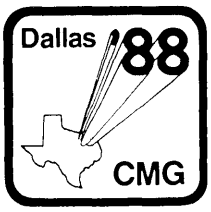
An Intuitive Approach To Performance Prediction with Application to Workload Management in VM SP/HPO, **Hellerstein,** 53.
A Flexible Approach To Performance Prediction For Large System Software Development, **Long,** 167.

A Systematic Approach For Identifying Peak Workload Intervals, **Artis,** 287.
A Capacity Planning Methodology That Is Destined To Succeed, **Lawrence,** 496.
Storage Performance, An Eight Year Outlook, **Goldstein,** 579.
Workload Characterization And Forecasting For A Large Commercial Environment, **Miller,** 655.
An Introduction to Business Element Forecasting, **Bowerman,** 703.
A Practical Use Of Natural Forecasting Units, **Reyland,** 710.
Systems Performance Management and Capacity Planning, **Domanski,** 725.
Fundamentals Of Statistical Workload Forecasting, **Artis,** 733.

Author Index

Allen, L.E. 630
Allison, G.B. 370
Amembal, S.P. 719
Arthurs, E. 959
Artis, H.P. 287,733
Bell, T.E. 549
Beretvas, T. 256
Bereznay, F. 40
Berry, M. 723
Bodley, P.G. 282
Bolker, E.D. 734
Bonnes, A.H. 72,102
Bowerman, J.R. 703
Brisson, L. 736
Burke, R.C. 880
Buzen, J.P. 246,929
Cahill, B.A. 201
Caldwell, S. 1
Chanson, S.T. 620
Cheeseman, J.T. 345
Cheng, M. 81
Christopher, D.C. 816
Claunch, C.V. 617
Cohen, E.I. 949
Comstock, C.L. 767
Cook, J.R. 636,822
Cooley, C.A. 714
Cunningham, D.L. 988
Day, B. 608
Deese, R.D. 764
Denis, R.H. 761
Dickerson, M.S. 118
Dines, W.F. 201
Domanski, B. 160,266,725
Duhl, B. 46
Emanuel, E.P. 970
Fahland, M.A. 936
Falk, A.M. 549
Friedman, E.M. 16,828
Futtrup, P. 122
Gantz, D.T. 405
Ghannad, H.H. 764
Gibson, J.L. 177
Gilbert, D. 87
Gillen, G.J. 546
Goldberg, R.P. 561
Goldenberg, K. 522
Goldstein, S. 579
Gonzalez, C.M. 405
Gray, W.T. 81
Guerrero, A.R. 596
Hackenberg, S.R. 252
Hackett, C.E. 33
Halladay, S. 719
Hanna, C. 237
Hardman, R.J. 184
Harmon, J.E. 858
Hellerstein, J. 53
Hellner, T.A. 415
Henley, M.R. 208
Henry, B.F. 546
Henry, D.J. 415
Houtekamer, G.E. 72,102
Ingraham, B.A. 504
Ingram, M.C. 177
Ingrassia, F.J. 151
Jack, H.E. 522
Jones, A.L. 672
Jones, K.C. 508
Ju, J. 1
Kaberon, D.A. 192
Kayser, L. 797
Keeney, G. 947
Keller, T.C. 336
Kirshin, M.F. 189
Kumar, A. 441
LaPlante, B.G. 936
Lawrence, J.D. 474
Lawrence, R.T. 496
Lee, M.R. 95
Lee, T.P. 452
Levy, A.I. 807
Levy, K.E. 217
Lichtig, B.A. 835
Lindsay, D.S. 310,330
Lipner, L.D. 561
Liu, I. 208
Lo, T.L. 446,464
Long, A.M. 167
MacKinnon, D. 272
Madison, H.G. 360
Major, J.B. 390
Marcus, M.B. 515
Martin Flickinger, G.K. 599
Martin, A. 736
Martindale, R.W. 835
McBride, D. 145
McCormack, B. 736
McDonald, T.W. 774
McElwee, M. 787
McNutt, B. 9,399
Merges, M.J. 64
Merrill, H.W. 222
Michlin, J. 736
Miller, G.W. 553,655
Moore, F.G. 685
Moore, M.L. 405
Mullen, J.W. 246,929
Mungal, A.G. 296,916
Murphy, R. 482
Murray, J.W. 9
Nearing, N.M. 432
Nelson, N. 574
Nelson, T.L. 936
Olberding, T. 741
Olcott, R. 749,889
Paans, R. 962
Panduro, M. 122
Parker, G.H. 651
Parmelee, T.L. 835
Peng, J. 464
Percell, J.W. 685
Pilch, J.P. 266,725
Place, J.P. 79
Pool, A.C. 901
Quernemoen, J.M. 936
Raghuraman, M.K. 441
Reyland, J.M. 710
Richardson, J.W. 538
Robertson, E.E. 988
Rosenberg, J.L. 490,531
Salminen, V.J. 566
Salsburg, M.A. 423
Samadi, B. 458
Samson, S. 290
Sannerud, B. 790
Sarna, D.E. 591
Scott, R.J. 801
Shane, J.G. 822
Shub, A.H. 110

Shum, A.W. 246,929
Simmons, G. 947
Smith, C.U. 138,991
Smith, H.L. 606
Smith, W.M. 177
Stancavish, J.J. 543
Stanton, J.A. 698,989
Steindel, G.C. 360
Stevens, D.F. 569
Streger, M.S. 795
Stuck, B.W. 959
Suhler, W.C. 167
Sutherland, J.C. 304
Taylor, M.L. 787
Temperly, J.F. 184
Terplan, K. 865
Thorley, T.M. 549
Tornheim, J.J. 351
Tron, A.W. 304
Van Hoboken, W.C. 72,102
Van Kinsbergen, J. 233
Verbil, H.E. 611
Vestermark, B 122
Wandzilak, J. 482
Weishar, D.J. 720
Wells, R.C. 586
Wessner, B. 518
Williams, B.E. 808
Williams, R. 522
Williams, R.T. 351
Wilson, D. 279
Wilson, D.D. 606
Wong, A.S. 620
Yount, M.W. 133
Zimmer, H. 981



PRELIMINARY CALL FOR PAPERS

CMG '88 INTERNATIONAL CONFERENCE

ON THE MANAGEMENT AND PERFORMANCE

EVALUATION OF COMPUTER SYSTEMS

December 12-16, 1988 **Loew's Anatole,** **Dallas, Texas**

It is not too soon to be thinking about the paper you are going to submit for presentation at the CMG '88 International Conference, held at the Loew's Anatole, Dallas, Texas. The Conference dates are December 12-16, 1988.

Although we're interested in receiving papers describing new ideas or techniques, we also want reports on experiences (either success or failure) in CPE areas. Papers, tutorials, new books, and BOF interest areas are encouraged in the following performance related areas:

Establishing a Capacity Management Program
Justification, Planning, and Control of Personal Computers in the corporation
Capacity planning and prediction
Management reporting
Workload characterization/forecasting
Distributed processing analysis
I/O subsystem performance analysis
User requirements and user service agreements
Office automation analysis
Installation management performance issues
UNIX
Artificial intelligence and expert systems
Establishing goals and measuring productivity
DP budgeting, cost accounting, charge-back, and financial management
Hiring and managing a CPE staff
Professionalism in the CPE field
Software performance: design, development, implementation, and production
Problem management and change control
Capacity management relationship to corporate staff
Network design and optimism

Preference will be given to papers and books that have not been previously published or presented. Authors of papers will be required to assign copyright for their papers to CMG for publication in the Conference Proceedings. It will be the authors responsibility to secure any release required by their employers for publication.

SCHEDULE

April 1, 1988	Abstracts due (10 copies)
June 1, 1988	Completed papers due (10 copies)
July 25, 1988	Notification of acceptance
Sept. 2, 1988	Final, camera-ready copy due

INQUIRIES

General Chairman, CMG '88
CMG Headquarters
6397 Little River Turnpike
Alexandria, VA 22312
(703) 354-3306

CMG OFFICERS AND DIRECTORS

PRESIDENT
Mel Boksenbaum
Amdahl Corporation
1250 E. Arques Avenue
P.O. Box 3470 MS 126
Sunnyvale, CA 94088
(408) 746-6761

VICE PRESIDENT
Marilyn T. Kanas
International Systems Services Corp.
Suite 201, 1 Greentree Centre
Marlton, NJ 08053
(609) 596-6700

TREASURER
Robert L. Morrison
IBM Corporation
Dept. 1P4 Bldg. 987
P.O. Box 390
Poughkeepsie, NY 12602
(914) 432-4713

SECRETARY
Jason Shane
Coldwell Banker Residential Group
nformation Services, Suite #200
23046 Avenida de la Carlota
Laguna Hills, CA 92653
(714) 859-5380

DIRECTORS

H. Pat Artis
Performance Associates, Inc.
2059 Kedge Drive
Vienna, VA 22180
(703) 281-7399

Thomas E. Bell
Rivendel Consultants, Inc.
Computer Technology Group
2921 Via Pacheco
Palos Verdes, CA 90274
(213) 377-5441

George W. Dodson
IBM Corporation
P.O. Box 152750 M/S 15-07-60
Irving, TX 75015
(214) 556-4714

Kenneth W. Kolence
Kolence Associates
3591 Louis Road
Palo Alto, CA 94303
(415) 856-4078

Dr. T. Leo Lo
McDonnell Douglas
P.O. Box 516 (WSH/306/3)
St. Louis, MO 63166
(314) 233-5393

G.W. (Bill) Miller
American Airlines
P.O. Box 582809 MD 353
Tulsa, OK 74158-2809
(918) 832-4986

J. William Mullen
BGS Systems, Inc.
25511 Longhill Lane
Spring, TX 77373
(713) 796-3000

TRANSACTIONS EDITOR
J. William Mullen
BGS Systems, Inc.
25511 Longhill Lane
Spring, TX 77373
(713) 796-3000

Copyright 1987 by the Computer Measurement Group, Inc. Copying is permitted without payment of royalty provided that (1) each reproduction is done without alteration, (2) the *CMG '87 Conference Proceedings* cover and copyright notice are included as the first page, and (3) that copies are used for educational purposes. If copying is done for commercial purposes, a royalty of $1.00 per page copied must be paid to the Computer Measurement Group, Inc., 6397 Little River Turnpike, Alexandria, VA 22312.